Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro sběr a analýzu požadavků dle metody FURPS

System for System Requirements Specification According to the FURPS Methodology

2015

Petr Bláha

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Bláha**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Systém pro sběr a analýzu požadavků dle metody FURPS

System for System Requirements Specification According to the FURPS Methodology

Zásady pro vypracování:

Cílem systému je podpora procesu sběru požadavků a jejich následná správa. Mezi základní funkce systému patří možnost vytvoření nového projektu v systému. Tomuto projektu bude uživatel přiřazovat jednotlivé požadavky a to dle metodiky FURPS. Požadavek může být rozšířen o další parametry, jako jsou například: priorita požadavku, verze požadavku, závislost požadavku na jiném, vytvoření exportu požadavků do textových editorů a odhad doby trvání daného projektu. Systém bude realizován formou internetové stránky, jejíž grafický návrh bude plně responzivní. Uživatel tak bude moci požadavky spravovat z přenosných zařízení (mobilní telefony,tablety a jiné).

Stěžejní body práce tedy jsou:
1. Vytvoření analýzy dané problematiky a návrh řešení projektu. V rámci návrhu i analýzy je nutné dodržet dobré zásady pro tvorbu softwarového projektu.
2. Implementace projektu, která bude zahrnovat výše zmíněné funkce správy požadavků. Student mimo jiné navrhne a vytvoří komponentu pro vykreslování grafů. Tato komponenta bude vykreslovat vzájemnou závislost jednotlivých požadavků. Uživatel bude schopen za pomocí grafu simulovat odstranění určitého požadavku popisovaného projektu (odstraněním vrcholu grafu). Komponenta v takovém případě vykreslí graf všech požadavků, které jsou na vypuštěném požadavku závislé.
3. Student vytvoří servisní vrstvu, která bude realizovaná za pomocí technologie WCF (windows communication fundation). Tato vrstva bude nabízet externí přístup k datům projektů, a to ověřeným uživatelům prostřednictvím webových služeb.
4. Nasazení projektu a předání kompletní dokumentace.

Seznam doporučené odborné literatury:

[1] Troelsen, Andrew. Pro C# and the .NET 4.5 Framework 6th edition. Washington : Apress, 2012. 1430242337.
[2] Wiegers, Karl. Software Requirements 2. Washington: Microsoft Press, 2003. 0735618794.
[3] Bishop, Judith. C# 3.0 Design Patterns. Indianapolis : O'Reilly Media, 2007. 978-0-596-52773-0.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Plucar**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *30. dubna 2015*

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Janu Plucarovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Systém vytvořený v rámci této práce je určen pro sběr softwarových požadavků od klienta na informační systém pro něj vytvářený. Měl by být dostupný skrze internetovou stránku. Vytvořený informační systém uchovává uživatele a jejich data. Na základě zadaných dat si uživatel může vytvořit graf, podle kterého pozná, které operace mají přednost.

Cílem práce je vytvořit funkční systém, který by byl uživatelsky přívětivý a jednoduchý. Sytém by měl být vytvořen za pomocí návrhových vzorů a měl by být responzivní vůči různým typům mobilních i nemobilních zařízení.

Klíčová slova

Informační systém; Uživatel; Program; Příkaz

Abstract

System developed in this work is intended for gathering of software requirements from the client at information system developed for him. It should be available through the website. The developed information system keeps user accounts and their data. The user is able to create a chart based on the entered data by which it recognizes what operations take precedence.

The aim of this work is to create a functional system that would be user-friendly and simple. The system should be designed using design patterns and it should be responsive to different types of mobile and non-mobile devices.

Key words

Information System; User; Program; Command

Seznam použitých zkratek

Zkratka	Význam
EAF	Enterprise Architecture Framework
IS	Informační systém
RUP	Rational Unified Process
UML	Unifed Modeling Language
LINQ	Language Integrated Query
EF	Entity Framework
WCF	Windows Communication Foundation
IIS	Internet Information Services
MVC	Model-view-controller
UI	User interface
API	Application programming interface
T-SQL	Transact Structured Query Language
SQL	Structured Query Language

Obsah

Úvod ..... - 1 -
1 Rozbor stávajících systémů ..... - 2 -
2 Postup vypracování ..... - 4 -
3 Analýza a návrh ..... - 7 -
3.1 Správa požadavků na systém podle metody FURPS ..... - 8 -
3.1.1 F – funkční požadavky ..... - 9 -
3.1.2 U – požadavky vhodnosti k použití ..... - 9 -
3.1.3 R – spolehlivostní požadavky ..... - 10 -
3.1.4 P – výkonnostní požadavky ..... - 10 -
3.1.5 S – požadavky na udržovatelnost softwaru ..... - 10 -
3.2 Scénáře případu užití (use-case scenario) ..... - 11 -
3.2.1 Use-case diagram ..... - 13 -
3.3 Návrh uživatelského rozhraní ..... - 14 -
4 Praktická část ..... - 16 -
4.1 Datová vrstva ..... - 17 -
4.1.1 Entity Framework ..... - 19 -
4.2 Logická vrstva ..... - 21 -
4.2.1 Windows Communication Foundation ..... - 21 -
4.3 Prezenční vrstva ..... - 22 -
4.3.1 Bootstrap ..... - 24 -
4.3.2 ASP.NET Model-view-controller ..... - 26 -
4.4 Graf ..... - 27 -
4.4.1 CanvasXpress ..... - 27 -
4.4.2 Gephi ..... - 29 -
Závěr ..... - 30 -
Použitá literatura ..... - 31 -
Seznam obrázků ..... - 32 -
Seznam příloh ..... - 33 -

Úvod

V této bakalářské práci je vytvářen software pro sběr softwarových požadavků, a to včetně popisu postupů pro správné vypracování softwarového produktu. V průběhu tvorby softwaru prochází systém několika fázemi. První fází je analýza, kterou provádí softwarový analytik. Pro analytiky jsou důležité systémy navržené pro správu softwarových požadavků. Systémy pro sběr softwarových požadavků umožňují analytikům získat a uchovat informace o požadavcích klienta kladených na výsledný produkt. Pro definování softwarových požadavků, které určují vlastnosti výsledného systému, se postupuje podle osvědčených metodik. Podle jedné z těchto metodik, konkrétně metodiky FURPS se člení požadavky na informační systém v této bakalářské práci.

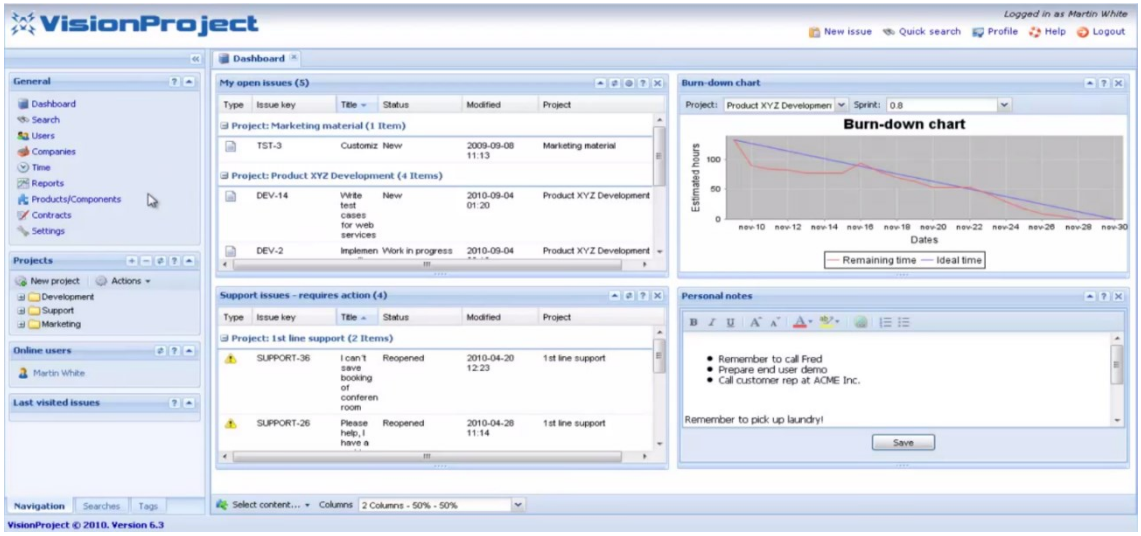
Cílem této práce je vytvořit systém pro sběr softwarových požadavků a jejich následnou správu. Systém je určen pro analytiky a uživatele zaměřující se na analýzu systémů. Mezi základní funkce systému patří registrace uživatele. Po přihlášení může uživatel vytvořit projekt, ve kterém bude moci zadat důležité údaje. Po vytvoření projektu bude uživatel vytvářet jednotlivé systémové požadavky, ke kterým může přidat vlastní popis či komentář. K obecné šabloně softwarového požadavku byl přidán atribut „počet hodin“. Podle tohoto atributu se bude vypočítávat odhadovaná cena projektu. Tato cena bude dána sumou hodin aktivních požadavků, násobenou výší ceny za hodinu, u vybraného projektu. Systém uživateli umožňuje přidávat k šabloně požadavku vlastní atributy. Uživatel si bude moci zobrazit systémové požadavky v grafickém zobrazení, kde budou vidět závislosti požadavků a jejich stav. Uživatelské rozhraní systému bude realizováno pomocí internetové stránky. Uživatel bude moci spravovat své projekty skrze přenosná zařízení s připojením k internetu, jako jsou mobilní telefony, tablety a další.

Práce je rozdělena do několika sekcí, z nichž hlavní jsou analýza, návrh, implementace a závěr. V sekci analýzy je řešena problematika sběru požadavků. Dále se zde uvádějí příklady řešených problémů např. v podobě FURPS požadavků nebo scénářů případů užití. V sekcích návrhu a implementace je nejprve zvolena vyhovující architektura, která je popsaná a zdokumentovaná na přiloženém příkladu. Dále jsou v této sekci řešeny dva architektonické problémy za pomocí návrhových vzorů. V poslední sekci je rozebrán závěr, kde je uvedeno shrnutí použitých materiálů a shrnutí problematiky řešené v této bakalářské práci.

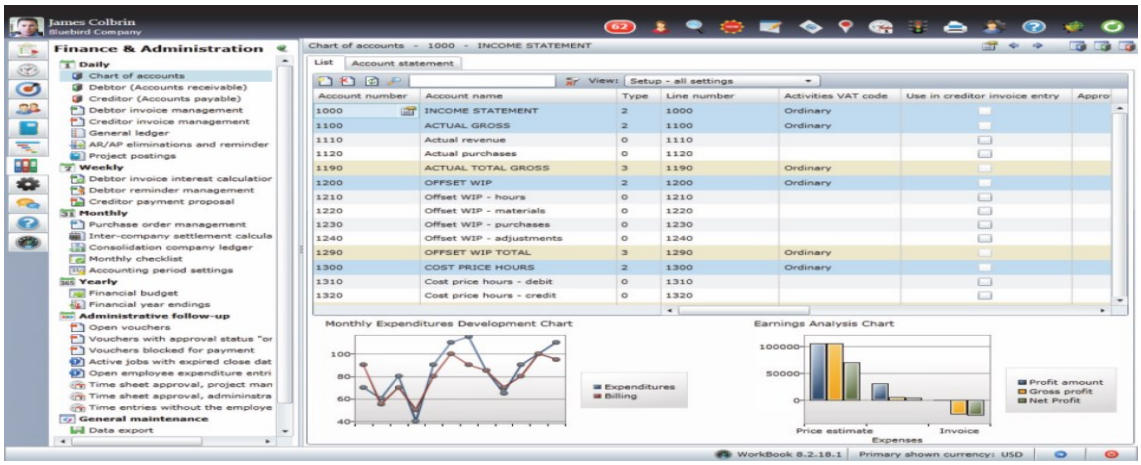
1 Rozbor stávajících systémů

Vyvstává otázka: „Proč vlastně vytvářet nový systém, když už systémy tohoto zaměření existují?“ Odpověď je jednoduchá. Nelze vytvořit univerzální systém, tak aby vyhovoval všem uživatelům. Například systémy na obrázcích č. 1 a 2 nemusí vyhovovat uživatelům, kteří nejsou obeznámeni s prací na podobných systémech, protože tyto systémy jim mohou připadat nepřehledné. Dalším problémem je nadbytečná funkčnost systémů, jenž lze pozorovat na obrázku č. 3, kde je vyobrazena funkce pro práci v týmu. Pro samostatné uživatele je možnost pracovat v týmu zbytečná. Pokud tedy uživatel pracuje sám a systém je placený (systém na obrázku č. 3 je placený), uživatel zbytečně platí části systémů, které nevyužívá. V některých případech je systém dostatečně jednoduchý a intuitivní jako například na obrázku č. 4, ovšem i zde systém nabízí desítky nepotřebných funkcí. Jelikož je většina systému pro sběr požadavků placená na základě dostupných funkcí, stávají se tyto systémy pro uživatele s malými nároky předražené vůči jejich využití. Další příklady systémů lze nalézt na webové stránce [1]. Vzhledem k nepřehlednosti systému, pak vznikají spory ohledně kvality systému a jeho budoucího využití. Z těchto důvodů se vytváří stále nové systémy, ze kterých si uživatelé pak vybírají dle svých potřeb.

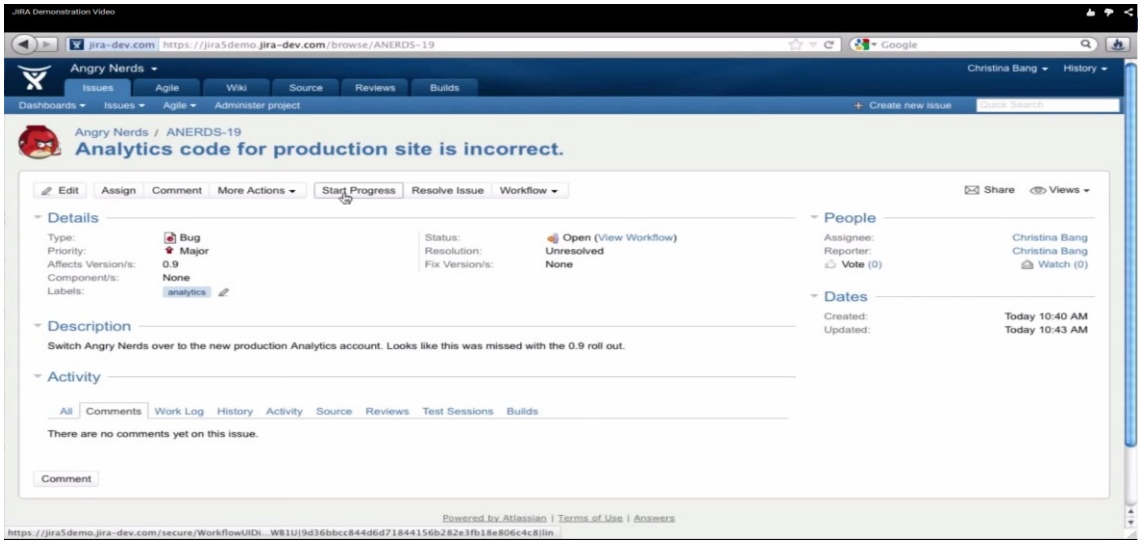
Cílem v této bakalářské práci je vytvořit co nejjednodušší a přehledné uživatelské prostředí, tak aby uživatel neměl moc přebytečných informací na obrazovce. K potřebným vlastnostem systému se bude muset „proklikat“. Počet potřebných kliknutí by neměl překročit určitou mez, protože se pak systém stává uživatelsky nepřívětivý. V případě systému implementovaného v této bakalářské práci je tento počet uměle stanoven na 4.



Obrázek č. 1 Systém VisioProject



Obrázek č. 2 Systém WorkBook



Obrázek č. 3 Systém JIRA

Obrázek č. 4 Systém PayPanther

2 Postup vypracování

Pro dodržení správného vývoje softwaru se zavádí takzvaný softwarový proces. Pro vypracování této práce je použit softwarový proces Rational Unified Process (RUP), který je následně krátce popsán.

RUP je objektově orientovaný iterativní přístup k životnímu cyklu systému. Cílem tohoto procesu je zaručit produkci kvalitního softwaru, který splňuje požadavky klienta, s předpověditelným časovým plánem a rozpočtem. Stěžejní fáze RUP jsou podle [2] tyto:

Zahájení (Inception)

V této fázi musí být ustanoveny používané prostředky. Například pro práci s analýzou se použije Unifed Modeling Language (UML). Dále musí být ustanoven rozsah vytvářeného softwaru. Pro ustanovení rozsahu je potřeba určit všechny vnější prostředky, se kterými bude software pracovat, například uživatelé nebo jiné systémy, a definovat tyto prostředky. To zahrnuje i identifikaci podnikových procesů a jejich popis za pomoci scénářů případů užití.

Výsledkem této fáze by měl být dokument pojednávající o potřebách softwaru, hlavních funkcích, počátečních modelech případu užití (rozpracován z 10-20%) a další.

Příprava (Elaboration)

Cílem této fáze je analýza doménové problematiky, ustanovení architektury, vytvoření softwarového plánu a odstranění vysoce rizikových částí softwarového návrhu. Ke splnění úkolu je třeba mít tzv. kompletní přehled o systému, i když to v této situaci není zcela možné (podle rozsahu softwaru). Architektonická rozhodnutí musí být tvořena s porozuměním vzhledem k celému systému neboli jeho rozsahu a funkčním/nefunkčním požadavkům. Výsledek této fáze rozhoduje o dalším postupu, jestli nějaký bude.

Výsledkem této fáze je model případu užití (alespoň z 80% dokončen), se všemi požadavky, účastníky a všemi případy užití kompletně zdokumentovanými. Včetně popisu softwarové architektury, spustitelným modelem architektury a souhrnem všech rizik atd.

Vypracování (Construction)

V této fázi se implementují všechny zbylé komponenty a funkce, které má požadovaný produkt poskytovat. Vypracování probíhá ve fázích, kdy na každém konci iterace by měla vzniknout funkční část tzv. demo verze. Tato demo verze se předvede zákazníkovi nebo vedoucímu vývoje softwaru a zkonzultují se změny popřípadě její nedostatky.

Na konci této fáze se rozhodne, jestli je vytvořený software připraven k nasazení vůči jeho rizikům. Většinou se takovýto software nasazuje jako beta verze.

Nasazení (Transition)

Účelem této fáze je nasazení výsledného produktu koncovým uživatelům. Jakmile je produkt nasazen u koncového uživatele, začíná fáze testování. V té dojde k identifikaci problémů,

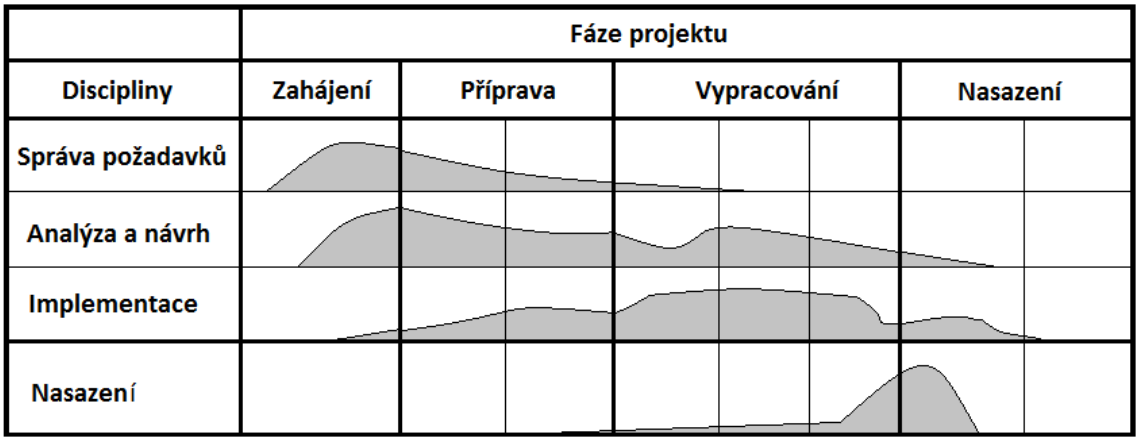
které vynutí vytvoření nové verze nebo dílčí aktualizace programu. Tato fáze může být časově náročná v závislosti na komplexnosti vytvářeného softwaru.

V tomto posledním bodě se rozhodne, zda byly splněny požadavky na software. V některých případech ukončení této fáze může znamenat začátek *zahájení.*

Použití RUP v rámci bakalářské práce:

Tato práce je vypracována podle obrázku č. 5, tedy softwarový proces je rozdělen na jednotlivé fáze, které se skládají z iterací. Jednotlivé fáze trvaly 14 dní a byly ukončeny konzultací s vedoucím práce.

Na obrázku č. 5 jsou na horizontální ose iterace daných fází. Jednotlivé fáze jsou rozděleny na iterace, přičemž po každé iteraci vzniká prototyp produktu, který je konzultován s vedoucím práce. Na vertikální ose lze vidět disciplíny, které jsou zpracovány. Oblast práce se postupně přesouvá s postupem času stráveným nad vývojem softwaru. Z obrázku vyplývá, že už v oblasti zahájení se začíná software implementovat a implementace pokračuje po dobu celého cyklu vývoje. Nasazení produktu v druhé iteraci už nebývá běžné, protože produkt je už nasazen jen se vydávají další verze.



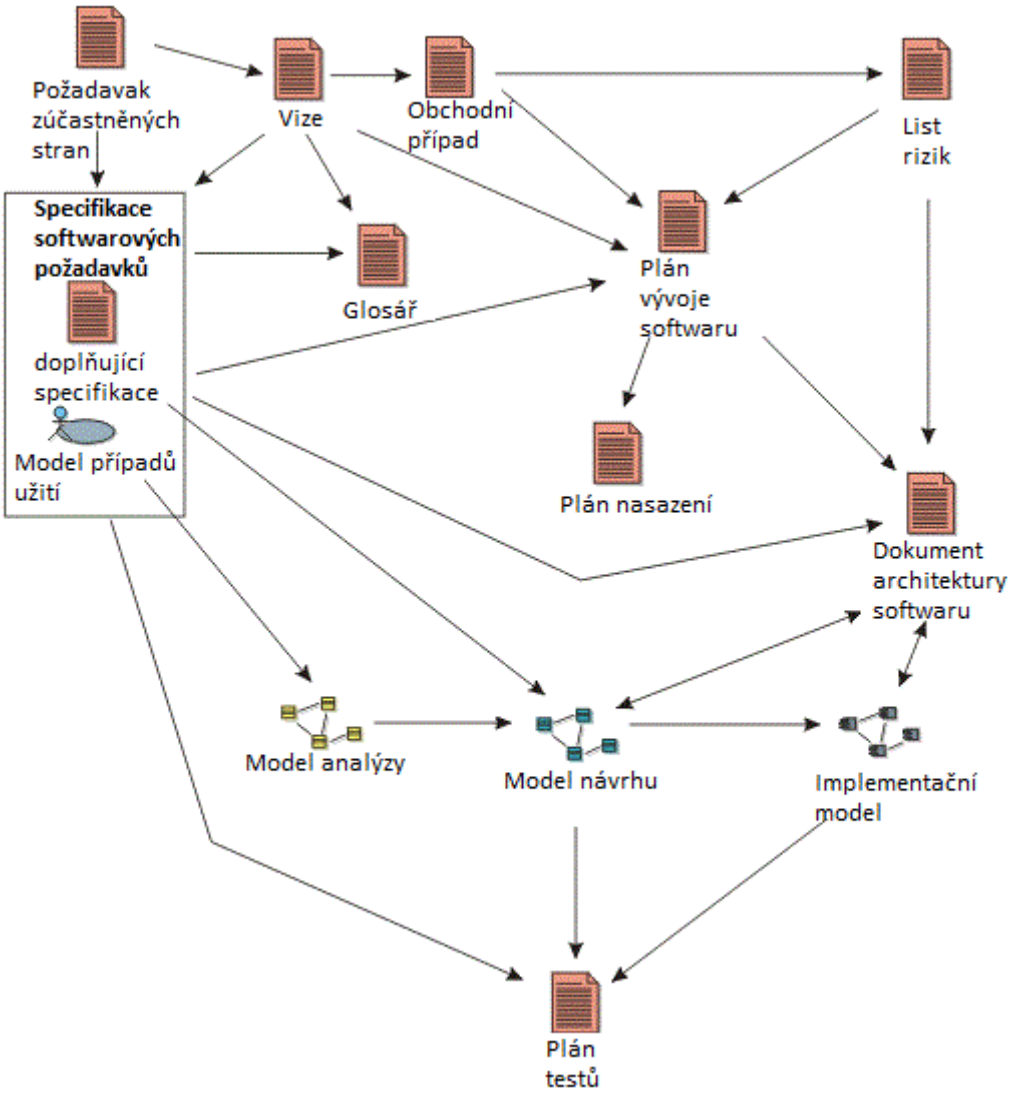
Obrázek č. 5 Fáze projektu

Při použití RUP vznikly takzvané artefakty (obrázek č. 6), konkrétně artefakt „vize“ a „specifikace softwarových požadavků“ podle [3], které jsou podrobněji rozepsány níže v textu.

Naskytuje se otázka: „Co jsou to vlastně artefakty a k čemu slouží?“ Artefakty jsou buď konečné pracovní produkty, nebo meziprodukty, které jsou vytvářeny a používány v průběhu projektu. Artefakty se používají pro zachycení a zprostředkování informací o projektu. Artefakty se dělí do kategorie „Dokument“ jako je například „Obchodní případ“ nebo „Dokument architektury softwaru“. Další kategorie jsou „Model“, do níž patří například „Model případů užití“ nebo „Model analýzy“. A kategorie „Model element“ do nichž patří části modelu v „Modelu“, například třídy nebo podsystémy.

„Proč použít jen dokument vize a specifikace softwarových požadavků?“ Protože další artefakty pro vytvoření takto malého systému nejsou zapotřebí. RUP se v plné míře využívá hlavně pro velké projekty. Upravením RUP lze získat dostatečnou podporu pro analýzu a vytvoření

softwaru malého rozsahu. Zároveň se po úpravě zachová dokumentace a přehled pracovního tempa. Jeden z dokumentů, který se využívá je dokument „vize“. V této práci byl použit pro upřesnění cíle systému a na jakou skupinu uživatelů a zástupců zúčastněných stran se tento systém zaměřuje. Konkrétní ukázkou sběru softwarových požadavků se přiblíží celková problematika bakalářské práce. Z tohoto důvodu byl vybrán dokument „specifikace softwarových požadavků“ jež se touto problematikou zabývá.



Obrázek č. 6 RUP artefakty

3 Analýza a návrh

V této kapitole je popsán dokument „vize“, který je rozebírán pomocí Enterprise Architecture Framework (EAF) konkrétně pomocí Zachmanova Frameworku (Obrázek č. 7). EAF slouží ke zpracování požadavků na systém, dá se upravovat podle požadavků uživatele.

V tomto případě toho, kdo provádí sběr požadavků od uživatelů navrhovaného systému. Zachmanův Framework je upraven k získávání požadavků jen pro jednu vrstvu.

	Z pohledu	Data	Funkce	Síť	Lidé	Čas	Motivace
Otázky		Co?	Jak?	Kde?	Kdo?	Kdy?	Proč?
Cíl/Rozsah	Projektanta	Seznam věcí důležitých v podniku	Seznam podnikových procesů	Seznam podnikových míst	Seznam důležitých organizací	Seznam událostí	Seznam podnikových cílů. strategií
Podnikový model	Vlastníka	Abstraktní údaje/ Navrhnout vzor	Podnikový procesový model	Podnikový logický model	Technologický postup	Rozvrh	Podnikový plán
Systémový model	Návrháře	Logický datový model	Systémová architektůra	Architektůra distribučního systému	Rozhraní mezi člověkem a architektůrou	Struktůra procesů	Model podnikových pravidel
Technologický model	Architekta	Třídní model	Model technologického návrhu	Technologická architektůra	Architektůra prezentace	Struktůra řízení	Návrh pravidel
Detailní zastoupení	Programátora	Definice dat	Program	Architektůra sítě	Architektůra bezpečnosti	Definice času	Pravidla dohadů
Fungující podnik	Uživatel	Požitelná data	Pracovní funkce	Použitelost sítě	Organizovanost funkcí	Realizovat plán	Pracovní strategie

Obrázek č. 7 Zachmanův framework

Jednotlivé otázky z obrázku č. 7 jsou následně podrobně vypracovány. Na otázku „**Co?“** je odpověď: **„**Systém pro získávání informací o požadovaném systému, které analytik/programátor musí získat pro navržení systému.“ Na otázku „**Jak?“** je odpověď**: „**Pomocí internetového rozhraní, kde bude v reálném čase moci zapisovat systémové požadavky a další věci potřebné pro analýzu projektu.“ Na otázku **„Kde?“** je odpověď**: „**Na mobilních (mobilní telefony, tablety, atd.) i nemobilních zařízeních s připojením k internetu a prohlížečem pro práci.“ Na otázku „**Kdo?“** je odpověď**: „**Analytici/programátoři, kteří potřebují systém pro ukládání požadavků získané od klienta.“ Na otázku „**Kdy?“** je odpověď**: „**Kdykoliv pomocí přenosných zařízení, která mají připojení k internetu.“ Na otázku „**Proč?“** je odpověď**: „**K ulehčení práce analytikům/programátorům se sběrem informací potřebných k vytváření aplikací nebo jejich částí.“

3.1 Správa požadavků na systém podle metody FURPS

„Co je to metoda FURPS a proč ji používat?“ Metoda FURPS byla původně vytvořena Robertem Gradym a rozšířena společností Rational Software. Metoda FURPS rozděluje softwarové požadavky do 5 kategorií. Tyto požadavky jsou nezávazně ohodnoceny podle měřítka vytvořeného pro daný software. Pro tento systém jsou určeny 2 stavy požadavku a to „Realizováno“ a “Nerealizováno“. Pokud je požadavek realizován, pak je ohodnocen hodnotou 1, v opačném případě je ohodnocen hodnotou 0. Na konci analýzy se sečtou hodnoty všech softwarových požadavků v dané kategorii a vydělí se celkovým počtem softwarových požadavků v této kategorii. Tento výpočet se provede pro všechny kategorie a průměrem výsledků vynásobeným stem se získá celková hodnota kvality softwaru v procentech. Takto se získá matematické ohodnocení vytvářeného softwaru. Kategorie pro rozdělení požadavků jsou popsány podle [4].

Pět částí metody FURPS znázorňují oblasti, do kterých se systém rozděluje.

Functionality (funkčnost): Funkční požadavky jsou považovány za architektonicky významné. Každý z těchto požadavků představuje funkci systému.

Usability (použitelnost): Jsou požadavky založené na otázkách uživatelského rozhraní, jako je dostupnost, konzistence a estetika.

Reliability (spolehlivost): Zahrnuje aspekty, jako je dostupnost, přesnost a využitelnost.

Performance (výkon): Zahrnuje propustnost, dobu odezvy systému, dobu zotavení, a dobu spouštění.

Supportability (podpora): Obsahuje řadu dalších požadavků, jako je snadné testování, přizpůsobivost, udržovatelnost, kompatibilita, konfigurovatelnost, škálovatelnost a další.

Níže je uvedena šablona pro zápis požadavku a příklady požadavků (zároveň část artefaktu „specifikace softwarových požadavků“, celý artefakt v příloze) na systém vytvářený v této bakalářské práci. Tato šablona bude implementována jako základní šablona v systému, kterou bude možno dále rozšiřovat. Po šabloně následuje ukázka výčtu požadavků systému.

Šablona softwarového požadavku

Označení: Jedinečný identifikátor požadavku. Může se skládat z prvního písmene kategorie, do které spadá a číselného pořadí požadavku.

Název: Krátký popis problematiky daného požadavku

Popis: Detailní popis problematiky daného požadavku

Priorita: Číslo označující důležitost daného požadavku ve stupnici od 10 po 1.

3.1.1 **F – funkční požadavky**

Označení: FP.1
Název: Uživatel musí být registrován pro práci s projekty
Popis: Uživatel bude automaticky přesměrován na přihlášení
Priorita: 9

Označení: FP.2
Název: Uživatel musí mít možnost zakládat projekty
Popis: Pokud je uživatel přihlášen musí mu být umožněno dostat se do sekce, kde může vytvořit projekt
Priorita: 10

Označení: FP.3
Název: Uživatel musí mít možnost přidávat požadavky
Popis: Uživatel po vybrání projektu bude mít možnost přidání nového požadavku
Priorita: 5

Označení: FP.4
Název: Přidání položek do projektu
Popis: Uživatel přidá do systému položky pod projekt
Priorita: 10
Závislost na požadavku:FP.2

Označení: FP.5
Název: Uživatel musí mít možnost grafického zobrazení
Popis: Uživatel po vybrání projektu bude mít možnost si nechat zobrazit vazby mezi požadavky
Priorita: 5
Závislost na požadavku:FP.3

3.1.2 **U – požadavky vhodnosti k použití**

Označení: UP.1
Název: Vytvoření přehledného uživatelského rozhraní (UI)
Popis: Vytvoření UI, které bude pro uživatele intuitivní a nad jednotlivými úkoly nebude muset ztrácet čas přemýšlením
Priorita: 6

Označení: UP.3
Název: Vytvoření plně responzivního UI
Popis: Uživatelské prostředí by se mělo přizpůsobit různým typům obrazovek nebo rozlišení
Priorita: 8

3.1.3 **R – spolehlivostní požadavky**

Označení: RP.1
Název: Systém musí ihned ukládat provedené změny
Popis: Systém po potvrzení akce musí uložit změny na server
Priorita: 9

Označení: RP.2
Název: Odolnost vůči chybám
Popis: Systém musí mít ošetřeny všechny vstupy
Priorita: 9

3.1.4 **P – výkonnostní požadavky**

Označení: PP.1
Název: Systém musí být schopen obsluhovat 500 uživatelů najednou
Popis: Při největším náporu systém musí být schopen plně obsloužit 500 uživatelů
Priorita: 8

Označení: PP.2
Název: Odezva serveru do 1.5 s
Popis: Maximální odezva při plném zatížení
Priorita: 8

3.1.5 **S – požadavky na udržovatelnost softwaru**

Označení: SP.1
Název: Podpora pro všechny prohlížeče
Popis: Systém se musí chovat a vypadat stejně v jakémkoli prohlížeči, ve kterém se s ním pracuje
Priorita: 8

Označení: SP.2
Název: Systém by měl být testovatelný
Popis: Systém by měl umožňovat programátorovi spustit automatické testy při zásadní změně doménové logiky
Priorita: 9

3.2 Scénáře případu užití (use-case scenario)

Dalším bodem analýzy je rozdělení systémových funkcí a přidělení těchto funkcí k jejich „aktérům“. V případě této práce je jediný uživatel, který má přístup ke všem podstatným funkcím. K vizualizaci vazeb mezi uživatelem a funkcemi slouží diagram použití, který je uveden níže na obrázku č. 8. Existují dva přístupy pro postup při vytváření scénářů a diagramu. Jeden z přístupů je nejdříve vytvořit seznam scénářů případů užití a následně z tohoto seznamu sestavit diagram. Tento přístup je použit v této bakalářské práci. Dalším je opačný přístup, neboli vytvoření diagramu a z něj vytvoření seznamu scénářů případů užití. Jelikož diagram případů užití (obrázek č. 8) ukazuje, jaké funkce systém obsahuje a který uživatel má přístup k daným funkcím, tak se dá vyčíst pouze název funkce, nikoliv její přímá funkčnost. K tomu slouží scénář případů užití, který nemá pevně danou podobu, ale může být realizován ve formě tabulky nebo prostého textu. Scénář případu užití je doplňkem diagramu případu užití a popisuje plný průběh dané funkce. Dále jsou zobrazeny příklady scénářů případů užití a diagramu případu užití vztahující se k této práci. Níže uvedená šablona bude implementována jako šablona v systému, kterou bude možno dále rozšiřovat.

Šablona scénáře případu užití

ID	Zkratka Use-case (Uc) a jedinečné číslo
Title (Nadpis)	Krátký a výstižný text nastiňující problematiku scénáře
Description (Popis)	Detailnější rozbor problematiky daného scénáře
Actor (Uživatel)	Osoba, která pracuje s problematikou daného scénáře
Pre-conditions(Předběžné podmínky)	Podmínky, které musí nastat, aby mohl scénář započít
Post-conditions(Konečné podmínky)	Podmínky, které musí nastat, aby mohl scénář být dokončen
Dependence (Závislost)	Závislost na některém z uvedených Uc
Main scenario (Hlavní scénář)	Hlavní linie, podle které by měl vypadat postup událostí
Exceptions(Vyjímky)	Události, které by mohli přerušit hlavní scénář a jejich řešení
Alternatives(Jiné možnosti)	Běh událostí, který může nastat místo hlavního scénáře

Příklad jednoduchého scénáře případu užití

ID	Uc.2
Title	Přihlášení do systému
Description	Uživatel před prováděním jakékoli akce se musí přihlásit do systému
Actor	Uživatel
Pre-conditions	Uživatel musí být registrován
Post-conditions	Uživatel musí být přihlášen
Dependence	Registrace
Main scenario	1) Uživateli se zobrazí přihlašovací formulář 2) Přepnutí uživatele do přihlašovacího formuláře 2) Ověření zadaných informací 3) Přepnutí přihlášeného uživatele na hlavní stránku
Exceptions	3a) systému se neshodují informace zadané uživatelem s informacemi zadanými klientem 3b) systém vyhodí chybovou hlášku
Alternatives	2a) systém si zapamatuje uživatele, není nutnost opakovat přihlášení

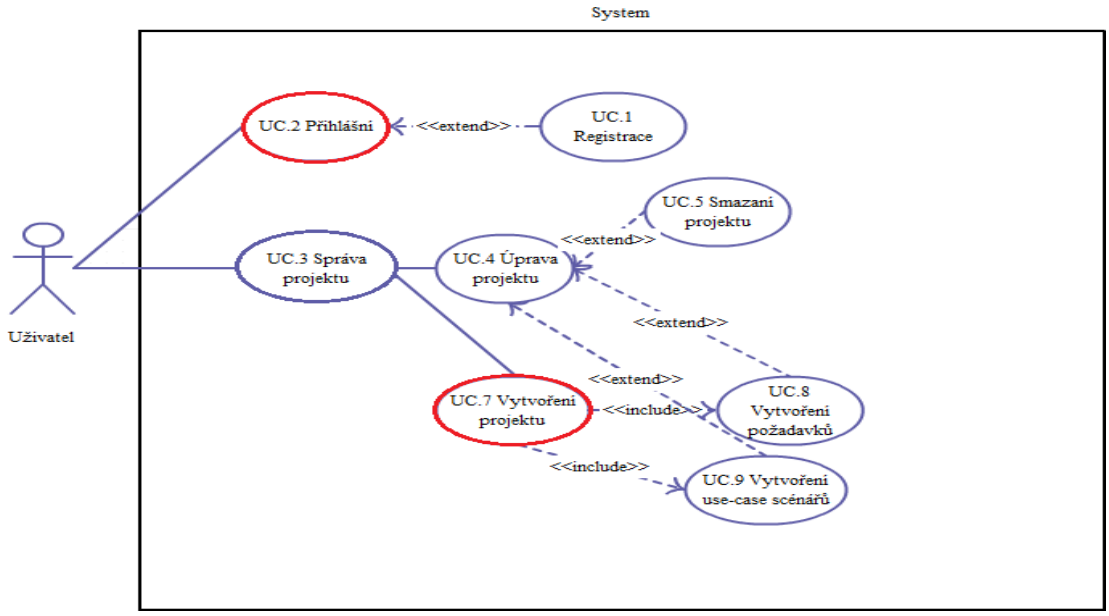
Příklad složitějšího scénáře případu užití

ID	Uc.7
Title	Vytvoření projektu
Description	Vytvoření projektu a dále umožnění jeho úpravy
Actor	Uživatel
Pre-conditions	Uživatel musí být přihlášen do systému
Post-conditions	Projekt musí být zapsán v systému
Dependence	Uc.3
Main scenario	1) Uživatel zvolí možnost vytvořit projekt 2) Uživatel vyplní údaje 3) Systém zkontroluje zadané informace

	4) Systém doplní základní informace 5) Uživatel potvrdí založení projektu
Exceptions	1a) Systém objeví duplicitu jména projektu 1b) Systém zobrazí chybovou hlášku 1c) Uživatel bude muset přejmenovat projekt 3a) Systém objeví neplatné/nevyplněné údaje 3b) systém vyhodí chybovou hlášku 3c) Uživatel musí doplnit/opravit chybné údaje
Alternatives	2a) Systém podle šablony vyplní údaje 4a) Uživatel změní ručně zadávací údaje

3.2.1 Use-case diagram

Tento diagram znázorňuje chování systému z pohledu uživatele. Účelem je zobrazit jednoduchý náhled nad funkcemi systému. Na tomto konkrétním obrázku č. 8 lze vidět, že uživatel má přístup k scénáři Uc.2 neboli přihlášení, který je rozšířen o scénář registrace, ale nemusí tomu tak být. Dále je na obrázku vidět, že uživatel má přístup ke správě projektu, která přímo obsahuje úpravu a vytvoření projektu. Vytvoření projektu je rozšířeno o vytvoření požadavků a use-case scénářů, na rozdíl od úpravy, která má jen přístup k těmto funkcím a ještě k funkci smazání projektu.

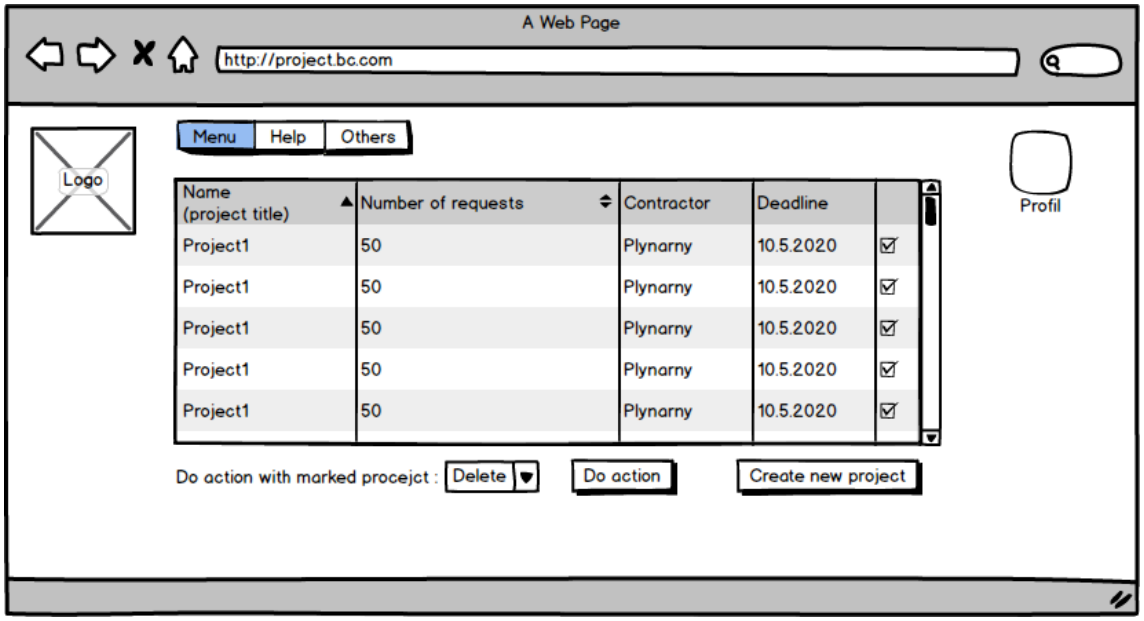


Obrázek č. 8 Use-case diagram

3.3 Návrh uživatelského rozhraní

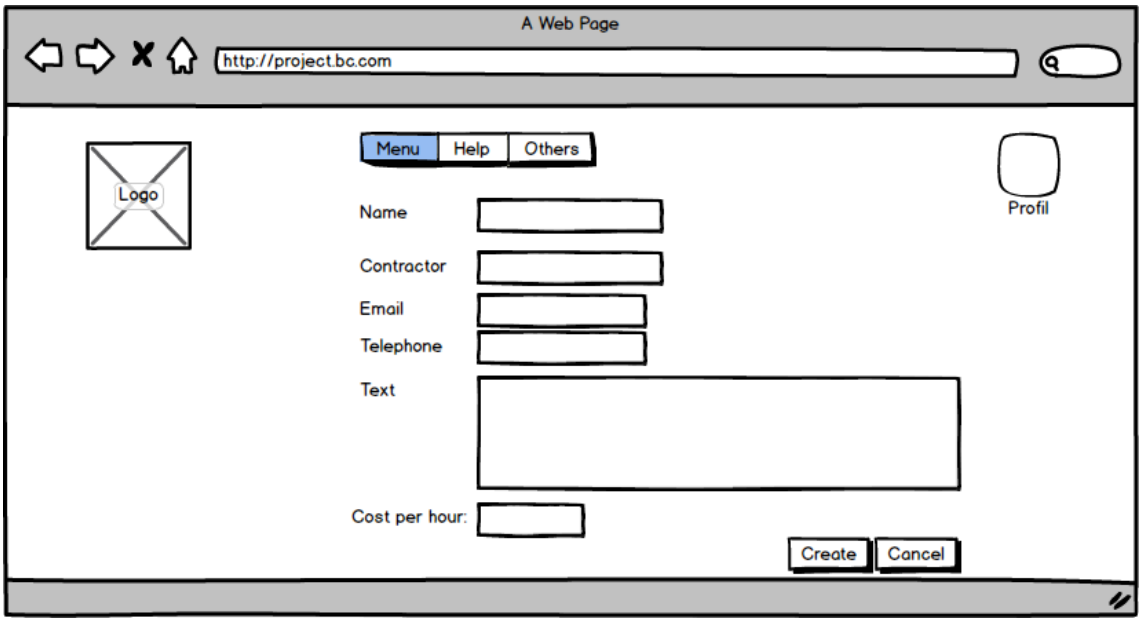
Pro znázornění vzhledu systému se klientům vytváří prototyp uživatelského rozhraní. Tento prototyp slouží ke znázornění budoucího vzhledu systému zákazníkovi, tak aby mohl vznést námitky nebo upřesnit požadavky, které na budoucí vzhled systému klade. Pomáhá při vypracovávání konečného vzhledu systému, ale není definitivní, je to jen předběžný návrh rozmístění komponent na viditelné části systému.

Obrázek č. 9 znázorňuje, jak by měly v systému vypadat formuláře pro hromadnou práci s projekty. Lze vidět seznam projektů a manipulační tlačítka na spodní části obrázku. Ve vrchní části je umístění hlavního menu pro pohyb na stránkách. Ikona loga profilu by měla reagovat buď na kliknutí, nebo na najetí kurzorem a měla by zobrazovat akce jako například odhlášení.



Obrázek č. 9 Souhrn projektů a práce s nimi

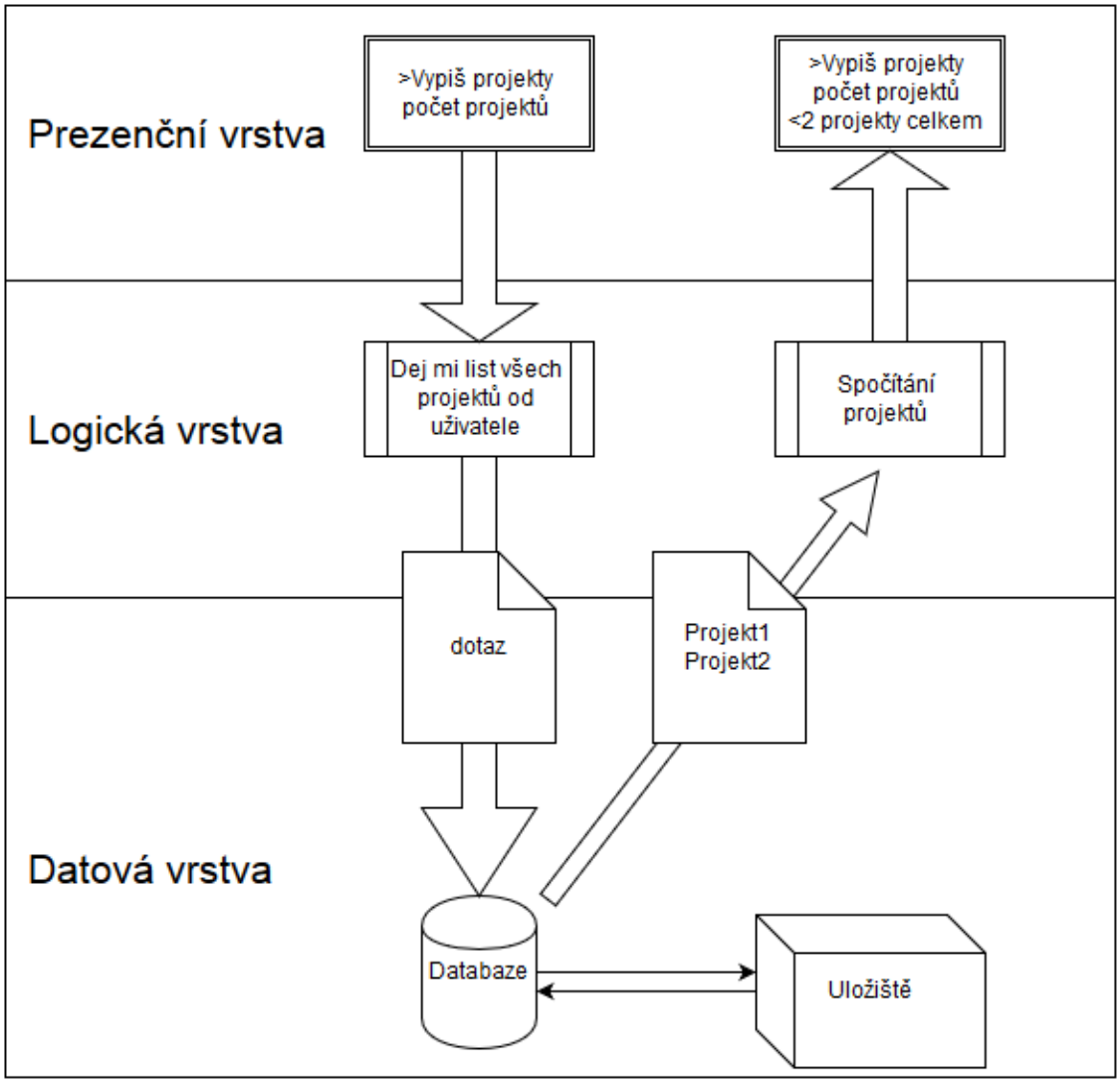
Obrázek č. 10 znázorňuje formulář pro vytváření projektu. Jedná se o formulář obsahující podobné rozložení jako předchozí model. Tento způsob rozmístění je zaveden z důvodu dobré orientace na stránce. Pokud by bylo rozložení stránky odlišné od ostatních stránek, uživatel by musel strávit čas hledáním ovládacích prvků.



Obrázek č. 10 Vytváření projektu

4 Praktická část

Analýza je tímto ukončena a přechází se k návrhu systému a popisu řešení problémů, které se při tvorbě Informačního systému (IS) vyskytovaly. IS je v této bakalářské práci rozdělen do tří vrstev podle principu N-vrstvé architektury. Naskytuje se otázka: „Proč vlastně rozdělovat systém do vrstev?“ Rozdělení systému do vrstev umožňuje zlepšit zabezpečení, jelikož na každé vrstvě je vlastní zabezpečovací algoritmus pro komunikaci mezi vrstvami. Po rozdělení se zjednodušuje možnost údržby, zlepšení a rozšíření systému. Každá vrstva může mít jeden nebo více zdrojů, se kterými komunikuje. Např. Prezenční vrstva může mít více zobrazení na každé zařízení (mobil, tablet, stolní počítač) a všechny zobrazení mohou komunikovat s jednou logickou vrstvou.



Obrázek č. 11 3-vrstvá architektura

První vrstva, tak zvaná prezenční, tedy nejvyšší vrstva, která je viditelná na obrázku č. 11 obsahuje definici uživatelského rozhraní, pomocí kterého uživatel komunikuje s IS. Hlavní funkcí této vrstvy je předání vstupů od uživatele do systému. Získaná data a požadavky jsou předána do logické vrstvy. Ta tyto podměty zpracuje a do prezenční vrstvy vrátí výsledky.

Druhá vrstva je nazývána logická vrstva. Vytváří spojení mezi sousedními vrstvami, přebírá informace, které dostane z prezenční vrstvy a zpracovává je. Pokud logická vrstva potřebuje data z datové vrstvy, tak pošle žádost do datové vrstvy. Výsledky poté použije pro výpočty nebo provede některou z potřebných akcí a předá je zpět do prezenční vrstvy. Logická vrstva zajišťuje výpočty, logická rozhodnutí a další funkce potřebné pro běh systému, které nejsou spojeny s přímou komunikací s uložištěm. Měly by se zde provádět veškeré akce, které nejsou přímo spojené s ukládáním a editací dat z uložišť nebo zobrazení výsledků uživateli.

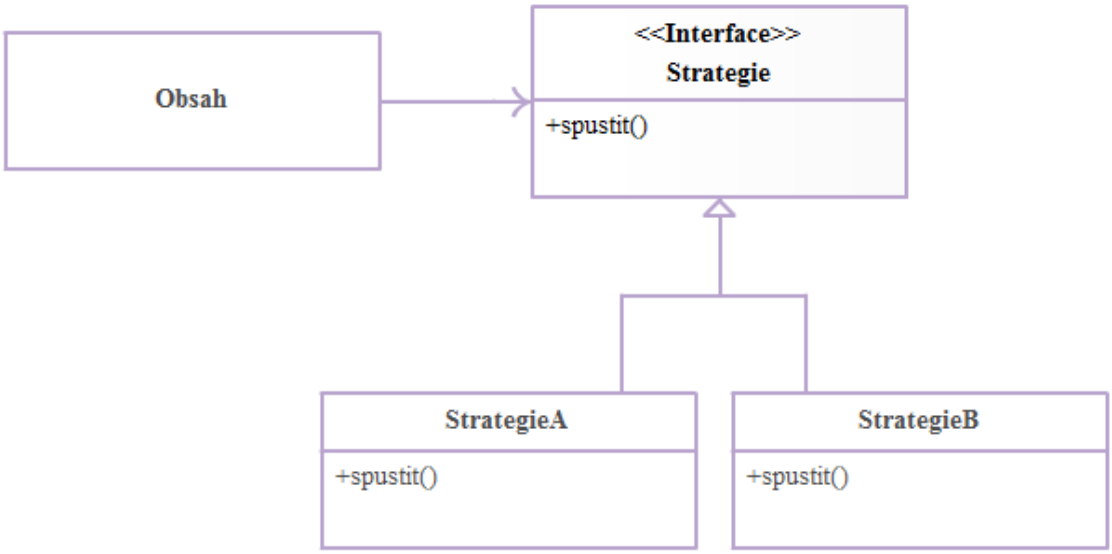
Poslední vrstva, tedy datová vrstva, pracuje s daty a jejich ukládáním, vyjímáním a kontrolou z uložišť jako jsou například databáze, soubory a další typy uložišť. V této vrstvě se může nacházet více druhů přístupů k uložištím dat, přičemž dané uložiště není závislé na výše uvedených vrstvách.

Vrstvy by měly být od sebe odděleny a jejich vytváření by nemělo být závislé na ostatních vrstvách. V následujících podkapitolách budou popsány problematiky jednotlivých vrstev. Pro přiblížení vývoje systému je popsána tvorba vrstev v takovém pořadí, v jaké byly vytvářeny.

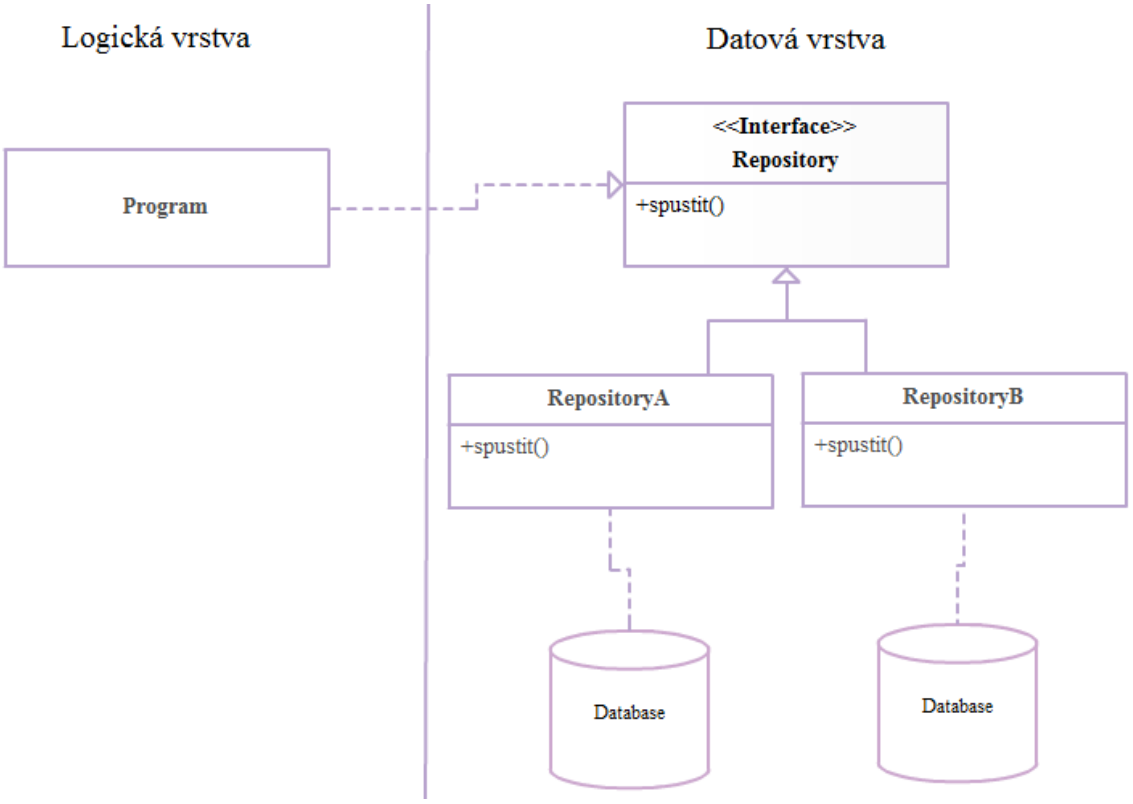
4.1 Datová vrstva

Prvním problémem bylo najít takové řešení, které by podporovalo vyměnitelnost zdroje dat. Výměna zdroje dat však nesmí zasáhnout do logiky aplikace, tedy volání datových metod musí zůstat stejné. Z důvodů rozšiřitelnosti datové vrstvy je zvolen návrhový vzor Strategy z publikace [5], který umožňuje rychle rozšířit nebo změnit používaný datový zdroj. Pro přístup k datům byl vybrán návrhový vzor Repository z téže publikace, protože dobře spolupracuje s návrhovým vzorem Strategy a umožňuje vytvořit mezivrstvu v datové vrstvě, která se stará čistě o přístup k datům, čímž zaručí, že práce s daty není přenášena na další vrstvy.

Níže uvedený obrázek č. 12 popisuje obecný návrhový vzor Strategy. Na tomto obrázku je vidět, že obsah přistupuje přes rozhraní strategie k přímým implementacím tohoto rozhraní. Neboli aby strategie mohla být použita obsahem, tak musí být rozšířena rozhraním strategie.



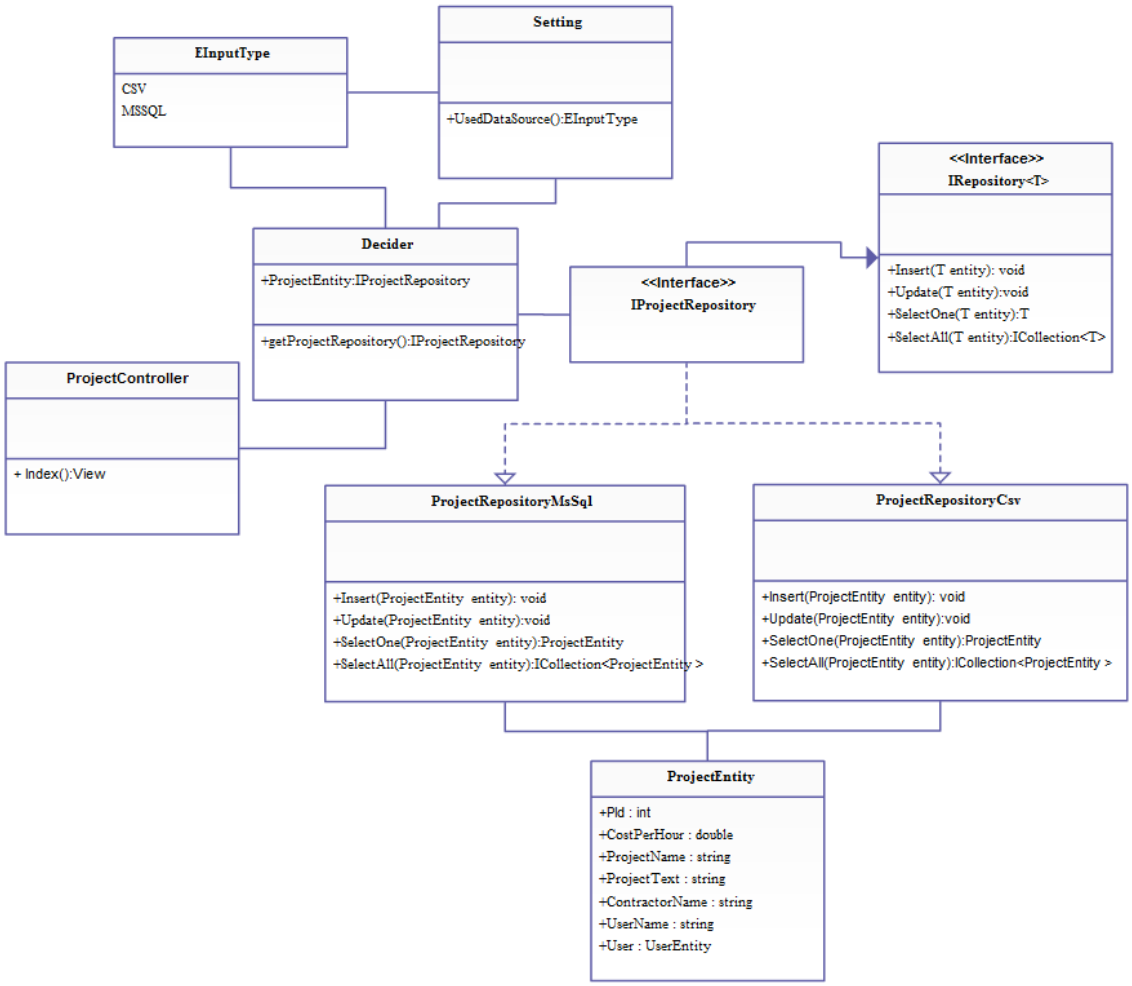
Obrázek č. 12 Vzor Strategy



Obrázek č. 13 Vzor Repository

Výše uvedený obrázek č. 13 popisuje návrhový vzor Repository. Na tomto obrázku je vidět podobná funkce jako na obrázku č. 12, ale u tohoto obrázku pracují konkrétní implementace repositářů nad databází nebo jiným datovým uložištěm.

Tyto dva výše uvedené vzory jsou zkombinovány a použity v IS. Pro vzor Repository je použito rozhraní (interface) v uvedeném příkladu se jedná o rozhraní Project. Toto rozhraní je použito pro základ vzoru Strategy. Poté třída Decider rozhodne, kterou strategii přístupu k datům zvolí, a to na základě proměnné ve třídě Setting.



Obrázek č. 14 Třídní diagram

Třídní diagram (obrázek č. 14) popisuje základní funkci datové vrstvy. Pokud se v ProjectControlleru zavolá metoda Index, tak tato metoda vytváří instanci Decideru aby získala repositář. Při vytváření Decideru se v konstruktoru rozhodne, na základě proměnné v Settingu, který repositář bude použit. Návratový typ metody vracející repositář je rozhraní Project. Aby metoda v Decideru mohla vracet repositář, musí konkrétní repositář implementovat rozhraní Project.

4.1.1 Entity Framework

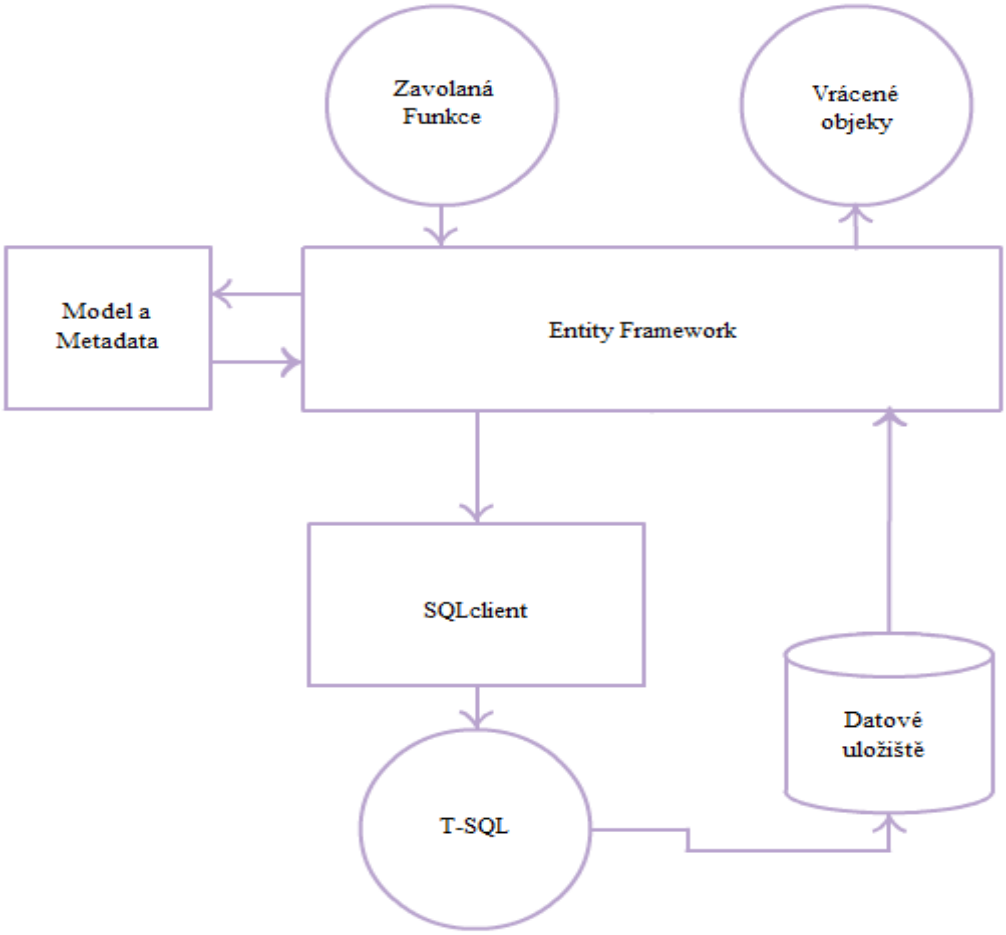
Entity framework (EF) je popsán v práci podle [6]. EF je objektově relační mapovač (object-relational mapper), který zprostředkovává vývojářům .NET práci s relačními daty použitím doménových objektů. Zjednodušuje psaní kódu pro přístup k datům, který vývojář musí

napsat. V původním ADO.NET bez podpory EF vývojář musel napsat kód, který mu umožňoval vybrat data z databáze. Tyto funkce se daly zjednodušit pomocí použití Language Integrated Query (LINQ), čímž se zkrátil kód. LINQ se bohužel prováděl nad daty přijatými z databáze a ne nad databází samotnou. Tento přístup nebyl výhodný, protože se musela vybírat nepotřebná data z databáze a tím docházelo ke zmenšení výkonu IS. EF dosahuje toho že LINQ se provede přímo nad databází samotnou a vrátí silně typové data. EF poskytuje služby jako například rozlišení identity, pomalé načítání (načítání jen části dat, které se zobrazují a ne všechny data z databáze).

Pracovat přes EF lze třemi různými způsoby. Prvním způsobem je způsob zvaný Code-First. Tento přístup umožňuje vytvořit databázi za pomocí doménového modelu, který je vytvořen nebo se bude vytvářet. Druhým je Database-Fist, kde vývojář vytvoří doménový model z modelu databáze. Tento způsob je efektivní, když se používá již existující databáze a je potřeba vytvořit doménový model, který bude použit pro vytvoření programu. Třetím typem je Model-First, kdy se ve vývojářském prostředí vytvoří model, podle kterého se automaticky vytvoří tabulky v databázi a třídy v programu.

Pro tvorbu programu v této práci byl použit první přístup neboli Code-First, který umožňuje vytvářet databázi pomocí migrací a příkazů příkazové řádky (Update-database, Enable-migration, Add-Migration).

Update-Database zkontroluje model v programu a změní nebo vytvoří databázi na předurčeném serveru. Pokud se má měnit již existující databáze, tak se musí povolit migrace klíčovým slovem Enable-Migration. Pro vytvoření nové migrace, která změní databázi lze použít klíčové slovo Add-Migration. Případně lze povolit automatické migrace v konfiguraci projektu. Tento krok umožní systému rozpoznat změnu modelu a pomocí příkazu Update-Database jí nahrát na server s databází a patřičně ji upravit. Pokud by se databáze měla radikálně upravit, přičemž by došlo ke ztrátě dat, musí se tato akce prosadit přidáním parametru -force nebo povolením automatické ztráty dat při migracích.



Obrázek č. 15 Entity framework

Na obrázku (Obrázek č. 15) je vidět jakou roli hraje EF v systému a kdy EF využívá LINQ příkazů. EF převede LINQ příkazy pomocí Structured Query Language (SQL) klienta do Transact Structured Query Language (T-SQL), T-SQL se následně provede nad databází a vrátí výsledky do EF, který je následně přepošle do systému. Model a metadata EF potřebuje, aby rozpoznal, ke kterému datovému uložišti se má připojit a ve kterých tabulkách hledat.

4.2 Logická vrstva

Tato vrstva je realizována za pomoci služeb Windows Communication Foundation (WCF), které umožňují systémům, aby metody z této vrstvy byly hostovány přes internet nebo na intranetu.

4.2.1 Windows Communication Foundation

WCF je rámec pro vytváření servisně orientovaných systémů popsaný podle této publikace [7]. Použitím WCF se mohou data posílat z jednoho koncového bodu servisní vrstvy na druhý. Koncový bod služby může být nepřetržitě dostupný na Internet Information Services

(IIS), nebo může být dostupný přes aplikaci. Koncovým bodem služby může být klientská aplikace, která zprostředkovává servisní koncový bod. WCF má rysy:

1. **Servisně orientovaná**

Architektura založená na službách se při vysílání a přijmu dat, spoléhá na služby. Výhodou služby (service) je, že je programována volně spřaženým stylem (loosely-coupled) na rozdíl od těžce programovaného stylu (hard-coded). Volné spřažení umožňuje lepší kontrolu nad změnami. Pomocí služeb lze poskytovat metody skrze platformy, tedy navazující systém se stává nezávislý na platformě.

2. **Servisní metadata**

WCF podporuje publikování servisních metadat za použití standardů jako jsou WSDL, XML schéma, WS-Policy. Tato metadata jsou automaticky vygenerována a nastavena pro přístup klienta k WCF službám. Metadata mohou být publikována skrze HTTP a HTTPS nebo použitím Web Service Metadata Exchange standardu.

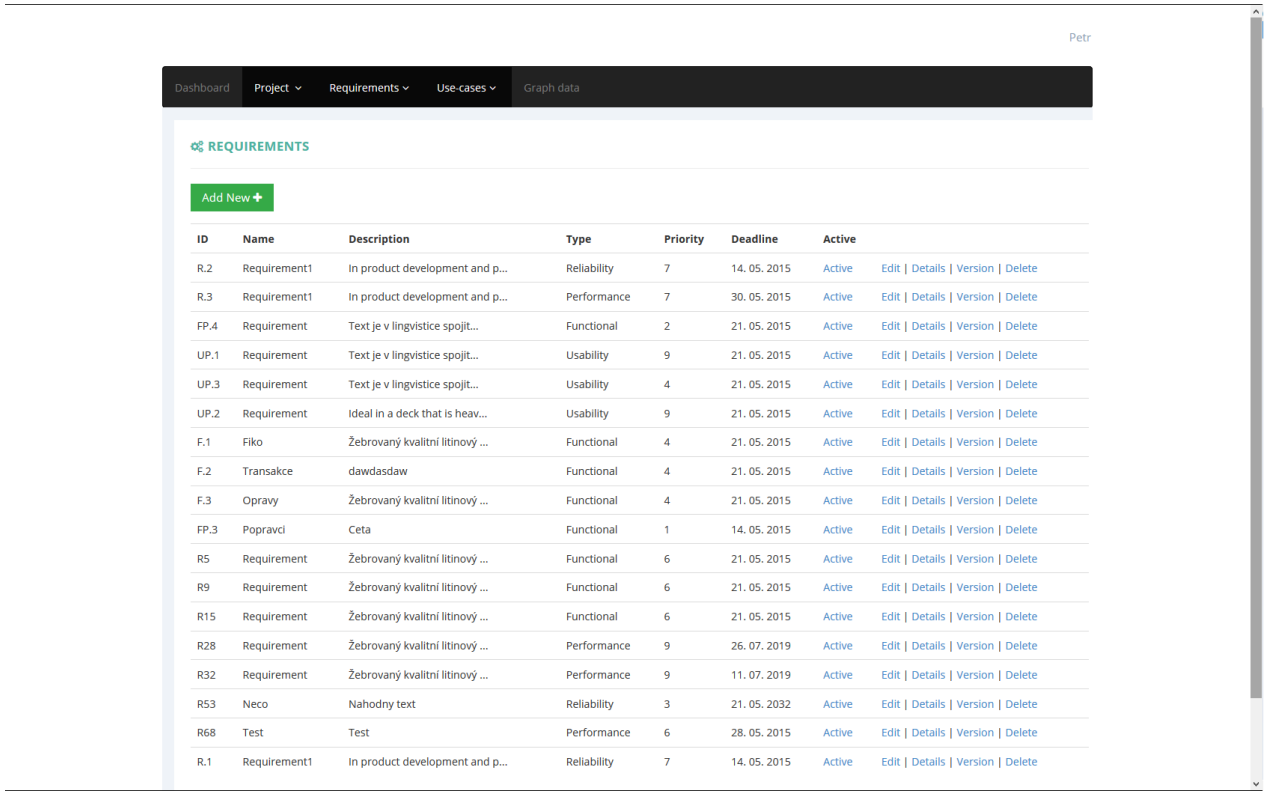
4.3 Prezenční vrstva

V prezentační vrstvě se kombinují dvě hlavní technologie rámec Bootstrap a architektura Model-view-controller (MVC). MVC vytváří nové rozvrstvení (pospáno níže v textu) a rámec bootstrap je použit pro zajištění responzivního vzhledu viditelné části systému. Tyto technologie jsou popsány níže ve jmenovitých podkapitolách.

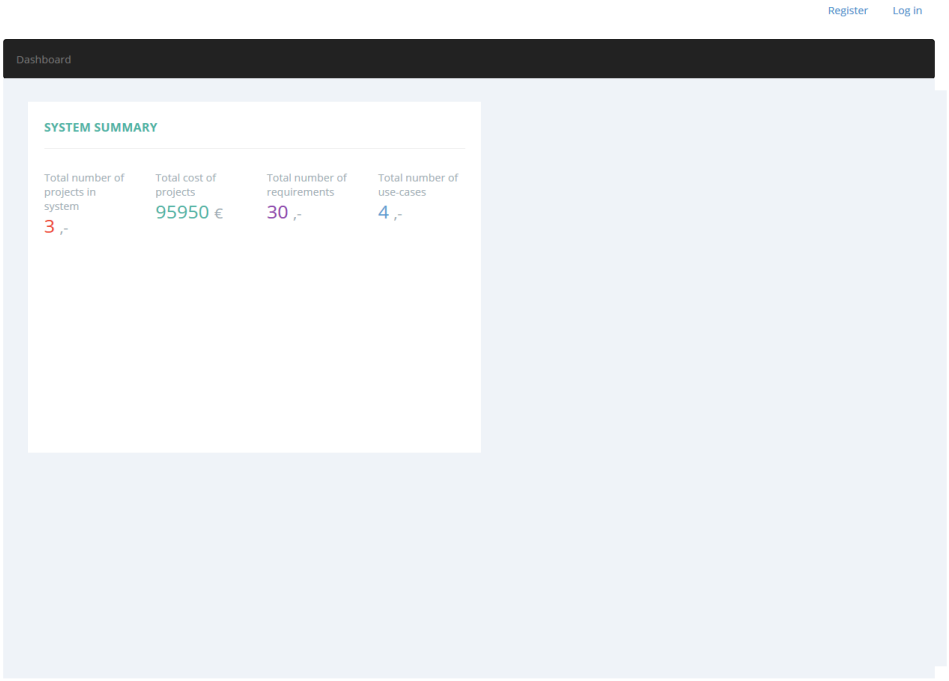
Kombinací použitých technologií vznikl systém zobrazený na obrázku č. 16. Na tomto obrázku lze vidět jednoduché uživatelské rozhraní s přehlednou tabulkou požadavků. Software na obrázku neobsahuje příliš mnoho funkcí ani tlačítek. Také není rozdělen do malých oken, které by obsahovaly dodatečné funkce jako například okno pro přidávání nového požadavku. Přidáním tohoto okna by způsobilo znepřehlednění tohoto systému. Ostatní požadované funkce jsou ukryty v pod nabídkách hlavního menu. Nabídky, které mají možnost podnabídek, jsou označeny tmavší barvou. Uživatel tak nemusí prozkoumávat, které nabídky mají podnabídky, což činí program příjemnější pro uživatele.

Přihlášeného uživatele lze vidět v pravém horním rohu na obrázcích č. 16 a 18. Provedení přihlášení je inspirováno většinou moderních systémů, a to formou interaktivního odkazu. Když se na tento nápis najede kurzorem, zobrazí se nabídka možností jak postupovat, včetně možnosti odhlášení uživatele. Jakmile není uživatel přihlášen (zobrazeno na obrázku č. 17), je tato část nahrazena tlačítky pro přihlášení a registraci.

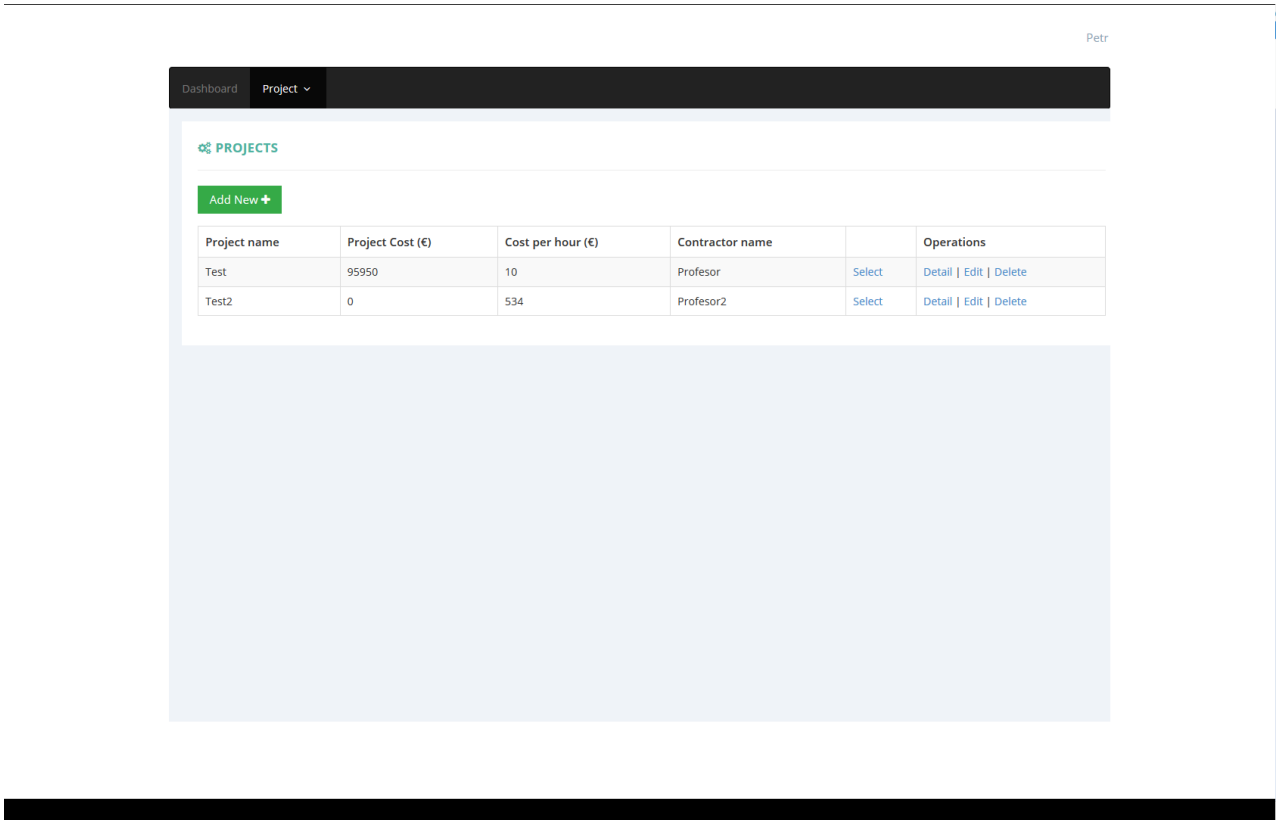
Nabídky, ke kterým uživatel v dané chvíli nemá přístup, jsou odebrány. Tento jev lze sledovat na obrázcích č. 17 a 18. Položky hlavní nabídky „Requirements“, „Use-cases“, „Graph data“, které jsou zobrazené na obrázku č. 16, jsou na obrázku č. 18 nepřístupné, protože není vybrán projekt. Tyto položky se zpřístupní po vybrání projektu. Samozřejmě je zamezen výběr více projektů zároveň.



Obrázek č. 16 Obrázek systému Requirements



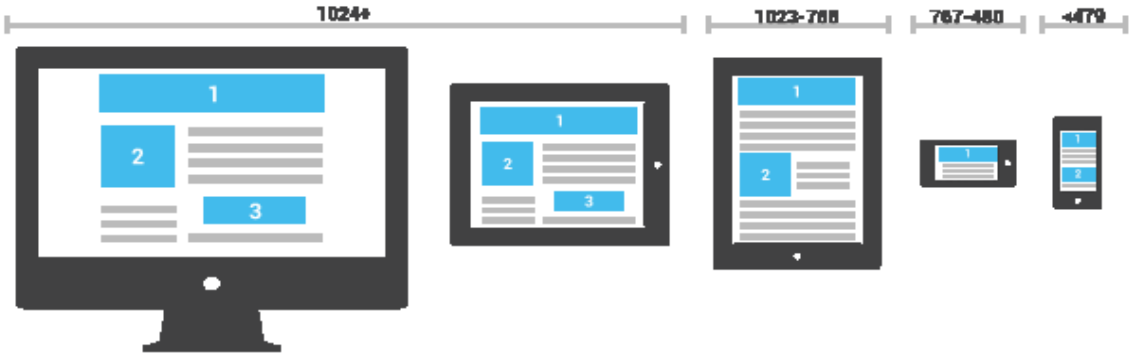
Obrázek č. 17 Obrázek systému Dashboard



Obrázek č. 18 Obrázek systému Projects

4.3.1 Bootstrap

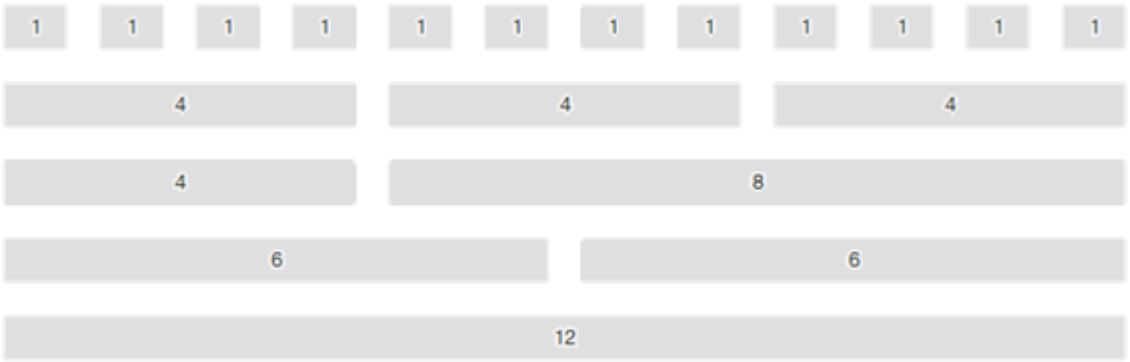
Rámec bootstrap podle [8] obsahuje mimo jiné CSS a HTML typografii, ikony, formuláře, tlačítka, tabulky, okraje a navigaci. Také obsahuje upravitelné jQuery-pluginy a podporu responzivního rozhraní. Výhodou rámce bootstrap je, že obsahuje velké množství volně dostupných nástrojů pro vytváření flexibilního a responzivního UI. Pomocí API se mohou vytvořit pokročilé komponenty, jako jsou například Scrollspy nebo Typeaheads, bez nutnosti psaní JavaScriptu. Další výhody rámce bootstrap jsou: dostupnost zdarma a podpora dominantních webových prohlížečů na trhu.



Obrázek č. 19 Responsivní prostředí

Výše uvedený obrázek (Obrázek č. 19) znázorňuje, funkci responsivního prostředí. Na obrázku lze vidět, jak se obsah stánky mění v závislosti na velikosti obrazovky. Při takové změně mohou jednotlivé části stránky změnit svojí pozici i velikost. Responsivnost není dána jen velikostí obrazu, ale také změnou velikosti okna, v němž se daná stránka nachází.

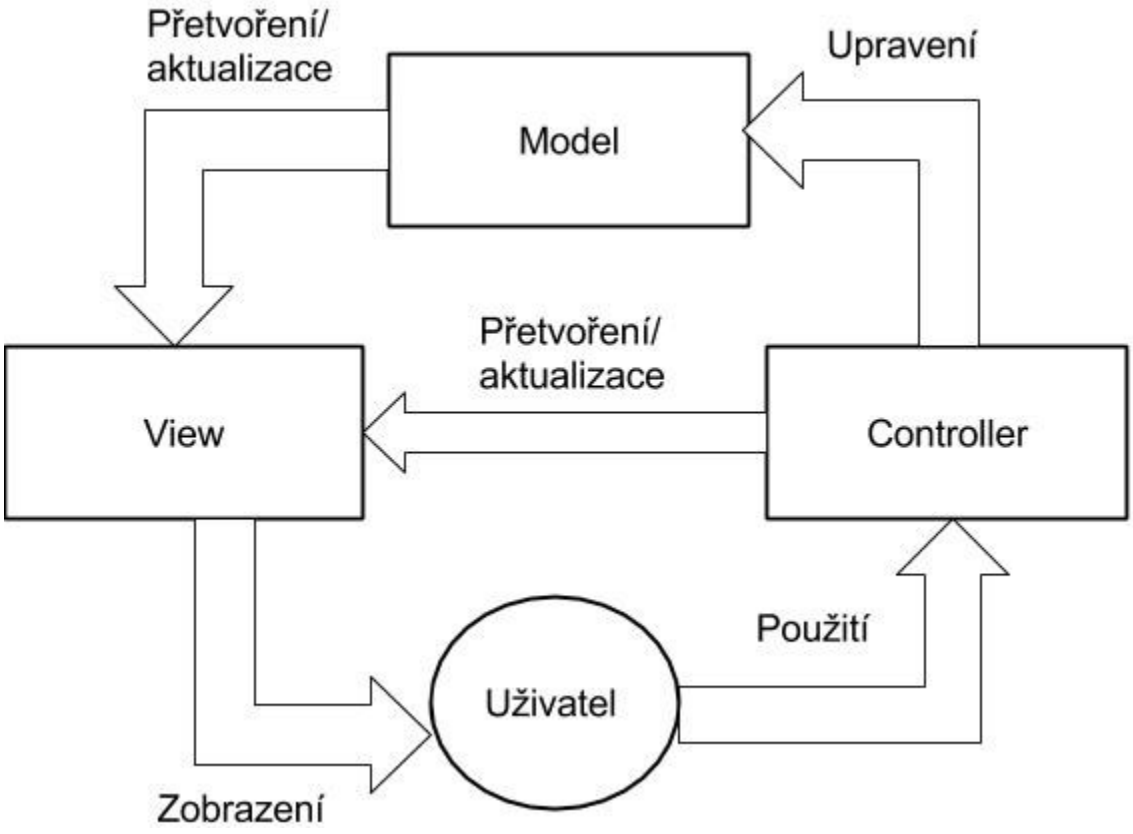
Rámec bootstrap funguje na základě responzivního prostředí a používá mřížku o dvanácti sloupcích (obrázek č. 20), do kterých rozděluje komponenty přidané vývojářem. Tyto komponenty se pak mění v závislosti na velikosti obrazu, jak znázorňuje výše uvedený obrázek (obrázek č. 19).



Obrázek č. 20 Mřížka bootsrapu

Tento rámec je použit na vytvoření responzivního UI, aby tlačítka, vstupy a výstupy z formuláře byly zarovnány a měnily se v závislosti na velikosti obrazu/okna.

4.3.2 **ASP.NET Model-view-controller**



Obrázek č. 21 MVC

Architektura MVC dělí aplikaci do částí Model, View, Controller. Jak je vidět na obrázku č. 21, pokud uživatel provede akci, která se mu vykresluje prostřednictvím view, tak se tato akce zpracuje prostřednictvím controlleru a je převedena dále. Pokud controller potřebuje provádět nějakou výpočetní činnost, tuto akci pošle modelu, který ji zpracuje a aktualizuje informace ve view. Jestliže se jedná o akci, pro kterou controller nepotřebuje provádět výpočetní činnost, jako například přesměrování, pak tuto akci provede sám. Až se dokončí akce provedená uživatelem, tak se tento úkon přenese do view, kde se zobrazí uživateli v jemu čitelné formě. Další rozdělení podle [9]:

Model - obsahuje doménovou logiku systému a je nositelem dat. Ve 3-vrstvé architektuře zasahuje do části datové a logické vrstvy.

View - je to viditelná část systému pro uživatele. Zprostředkovává vrstvu mezi systémem a uživatelem. Uživatelské akce se následně posílají do controlleru, který tyto akce zpracuje.

Controller - Obsluhuje view, které mu byli přiřazeny. Vytváří vrstvu mezi modelem a view. Řídí data, jenž se mají zobrazovat, a akce, které se mají provádět na základě logických operací určených vývojářem.

Problém MVC je jeho použití. Na první pohled není zcela patrné, kde jsou stanoveny hranice mezi doménovou logikou a logikou práce controlleru. Controller by měl fungovat jako spouštěč doménové logiky a neprovádět žádné akce typu výpočty, takovéto akce by se měly provádět ve vrstvě modelu. Controller by měl sloužit jako směrovač.

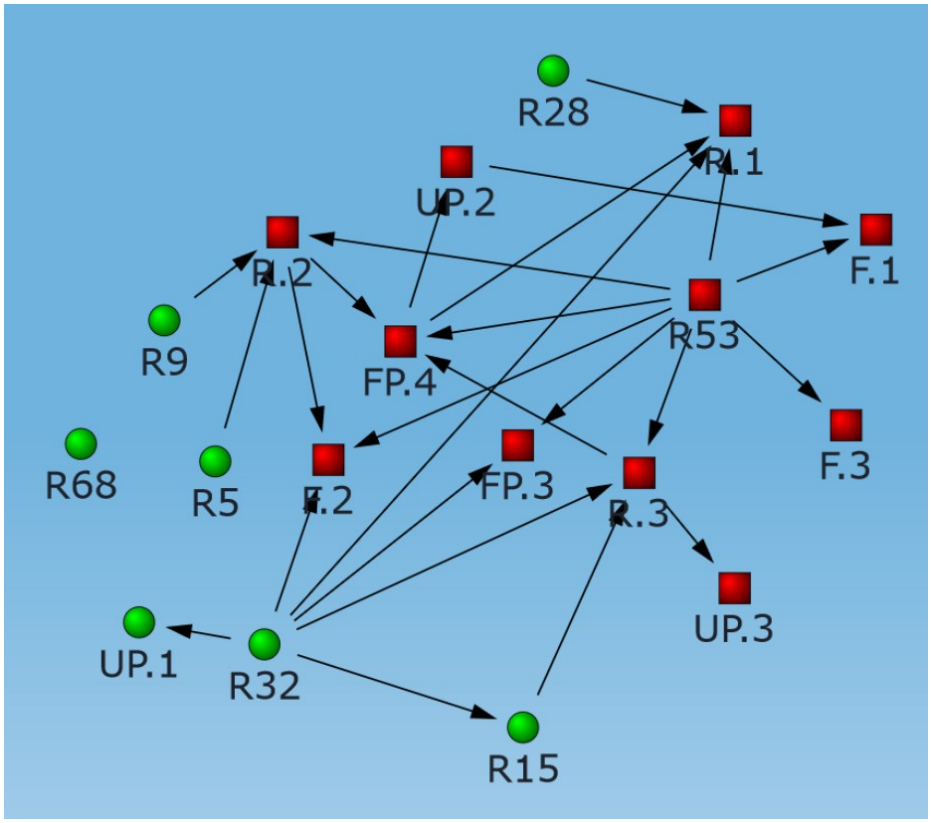
4.4 Graf

Pro zobrazení datové struktury přímo v prohlížeči je možno využít několik knihoven. Pro tuto práci byla zvolena knihovna CanvasXpress [10], která více vyhovuje potřebám zobrazení na rozdíl od např. Arbor.js [11]. Pro zobrazení by se daly využít také služby jako například Gephi services, které by přijaly data a vytvořily z nich graf. Bohužel tyto grafy by byly statické a těžce měnitelné, jelikož při změně požadavku by se musely znova sestavovat. V případě, že uživateli nebudou vyhovovat CanvasXpress grafy je vytvořena nástavba v podobě exportu dat do programu Gephi.

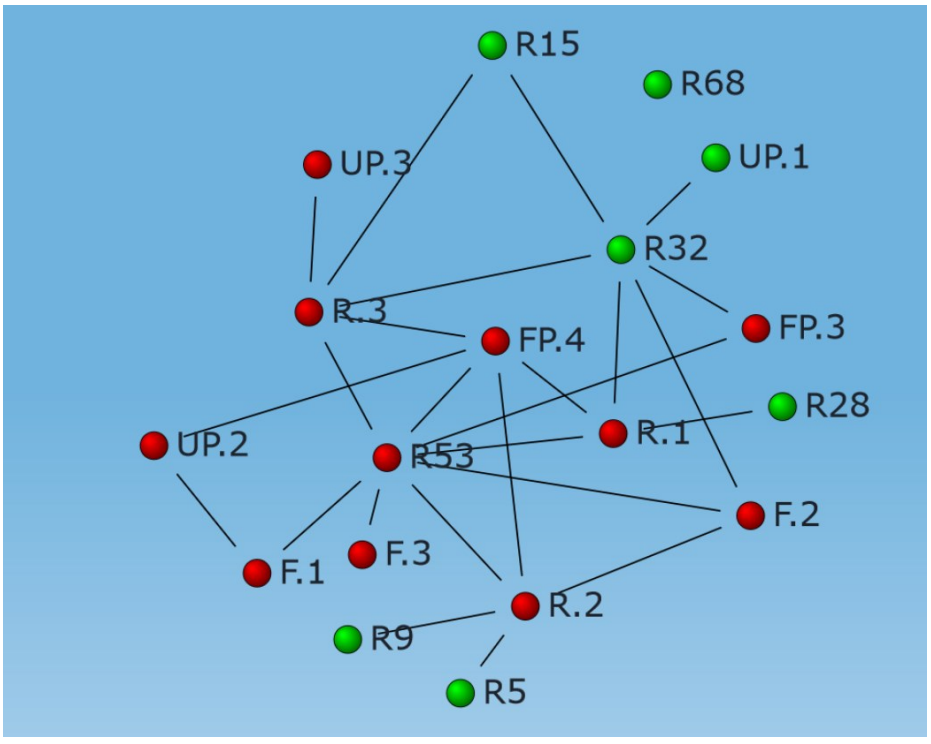
4.4.1 CanvasXpress

CanvasXpress je javascriptový nástroj pro podporu vykreslování grafu. Pomocí tohoto nástroje se dají vykreslit různé typy grafů. V této bakalářské práci je využit jen nástroj pro grafy typu „Network“, které se nejvíce podobají výslednému vzhledu grafu pro potřeby analýzy požadavků. Tento nástroj příjme data v JSON (JavaScript Object Notation) formátu, následně data rozdělí a vykreslí graf. CanvasXpress má mnoho nastavitelných vlastností, které lze najít v sekci dokumentace na [10]. V tomto bakalářském projektu jsou využity tyto vlastnosti: backgroundGradient1Color, backgroundGradient2Color, gradient, graphType, indicatorCenter, nodeFontColor, showAnimation, showNetworkNodesLegend, networkNodeMinDistance.

Systém vytvořený v rámci této bakalářské práce generuje dva typy grafů. První je zobrazen na obrázku č. 22, jedná se o orientovaný graf pro analýzu požadavků. Z tohoto grafu lze vyčíst, který požadavek je závislý na kterém požadavku. Druhý typ grafu je neorientovaný (viz obrázku č. 23), protože tento program je zaměřen na sběr požadavků a následné předvedení dat klientovi. Pokud klient uvidí, že významnou část systému nelze realizovat, z důvodu financí nebo technologií, může se rozhodnout tento systém nevytvářet. Oba grafy mají shodné zobrazení, kdy červené vrcholy reprezentují „Nerealizované“ požadavky a zelené vrcholy „Realizované“ požadavky. Vysvětlení pojmů „Nerealizované / Realizované“ požadavky je v kapitole 3.1.



Obrázek č. 22 Orientovaný graf z CanvasXpress



Obrázek č. 23 Nerientovaný graf z CanvasXpress

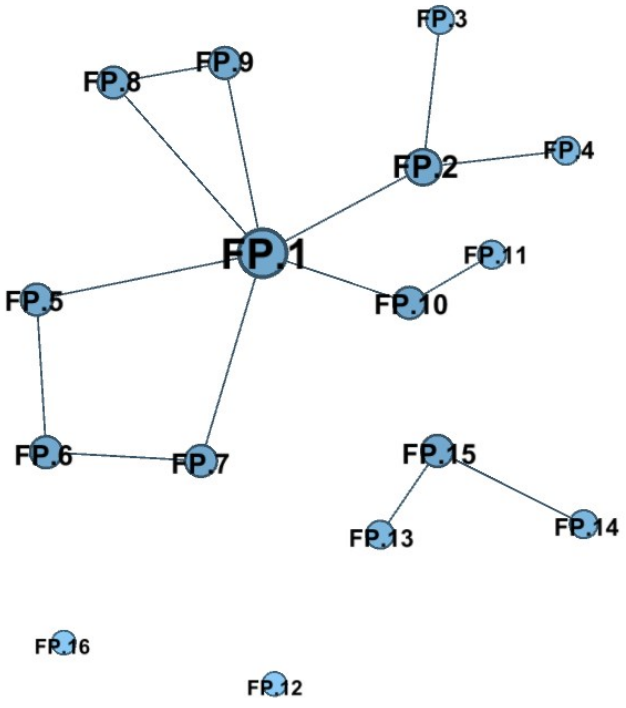
4.4.2 Gephi

Program Gephi je dostupný z uložiště GitHub viz [12]. Jedná se o interaktivní vizuální platformu pro všechny druhy sítí, složitých systémů, dynamických a hierarchický grafů. Je to nástroj pro uživatele, kteří se snaží nalézt informace v grafech. Uživatel manipuluje se strukturou, tvarem a barvami hran a vrcholů k odhalení skrytých vlastností. Cílem je pomoct analytikům při vývoji a při odhalování chyb softwaru. Jedná se o doplňkový nástroj při vytváření statistik.

Gephi dokáže zpracovávat formáty jako GEXF, GDF (formát použit pro tento systém), GLM, GraphML, Pajek Net a další. Gephi je založeno na OpenGL, umožňuje zobrazovat sítě do 50 000 uzlů a 1 000 000 hran.

Layout - algoritmus, který dává tvar grafu (na obrázku č. 24). Gephi umožňuje za běhu měnit vlastnosti grafu a tím zvyšuje schopnost poznání a učení z grafu. V layoutu se dají nastavit parametry, jako například rozlišení barev skupin, filtrace prvků, které mají nižší nebo vyšší počet hran než je žádáno. Ukazovat jména daných uzlů přiřazovat jim velikost podle počtu hran, se kterými daný uzel je spojen.

Pro tento graf (viz obrázek č. 24) je zvolen layout ForceAtlas2 a zapnuto barevné odlišení a úprava velikosti. Tyto vlastnosti závisí na počtu hran daného uzlu.



Obrázek č. 24 Ukázka grafu generovaného pomocí Gephi

Závěr

Tato práce je implementačního charakteru, proto byly na začátku práce porovnány stávající systémy, aby mohla být odůvodněna tvorba nového a zároveň aby byly eliminovány některé nedostatky ostatních systémů. Následně byl zvolen softwarový proces, podle kterého bylo postupováno při vytváření práce. Byly provedeny potřebné kroky při analýze, čímž byly určeny potřeby systému. V návrhu byla zvolena odpovídající architektura a návrhové vzory, vzhledem k potřebám rozvoje systému. Nakonec byl systém naimplementován a byly popsány použité technologie a konkrétní funkce systému.

Při vypracovávání této bakalářské práce bylo důležité se naučit pracovat se všemi výše uvedenými technologiemi a dalšími technologiemi, které nebyly v bakalářské práci použity jako například práci s SMTP nebo hašovacími funkcími.

V rámci této bakalářské práce byly splněny následující cíle:

- Vytvoření analýzy a návrhu řešení dané problematiky.

- Implementace projektu s požadovanými funkcemi, kdy graf je řešen pomocí vykreslování v canvasu a je vytvořen export pro nástroj Gephi.

- Implementace servisní vrstvy byla realizována pomocí WCF a zprostředkována přes webové stránky.

- Vytvoření dokumentace přiložené na disku

Externí nástroj pro vykreslování grafu byl zvolen z mnoha důvodů. Mezi ně patří velká podpora pro práci s datovými strukturami, poměrně velká komunita lidí, kteří jsou ochotni poskytnout informace pro vyřešení případného problému, rozsáhlá dokumentace k danému programu, jednoduchost daného externího programu a využití daného programu pro další funkce.

Pro budoucí vývoj projektu je možné přidat další požadavky na systém, které se musí analytik/vývojář dozvědět pro úspěšnou tvorbu výsledného produktu. Také by bylo možné přidat propojení s vývojovými systémy, kde by se vytvořený projekt promítl jako řešení (Solution). Pro každý požadavek by byla vytvořena podsložka, která by v budoucnu obsahovala vypracované řešení daného požadavku. Systém by mohl obsahovat podporu pro práci ve skupinách. Pomocí přidělování práv k danému projektu od tvůrce projektu nebo sledovacího systému pro klienta by bylo možné monitorovat, jak práce na projektu pokračují.

Vylepšením této práce by byla také implementace šablony pro generování smluv. Takto generovaná smlouva by se dala použít jako placená funkce zaměřená zejména na malé týmy nebo samostatně pracující programátory („freelancery“). Smlouva by obsahovala softwarové požadavky jako přílohu. Informace o účastnících smlouvy se mohou převzít z informací o projektu (při založení projektu se vyplní informace o zákazníkovi). Implementace převedení „aktivních“ požadavků do textové formy je v tomto systému již implementována, a to pomocí knihovny iTextSharp.

Použitá literatura

[1] Top Project Management Software Products. Capterra [online]. 2015 [cit. 2015-06-19]. Dostupné z: http://www.capterra.com/project-management-software/

[2] RUP – Rational Unified Process. Testování softwaru [online]. [cit. 2015-04-22]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/rup/

[3] Rational Unified Process: Artifacts. Rational Unified Process [online]. 2002 [cit. 2015-06-19]. Dostupné z: http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/ovu_arts.htm

[4] MALHOTRA, Namita a Pruthi SHEFALI. An Efficient Software Quality Models for Safety and Resilience. International journal of recent technology and engineering [online]. 2012, roč. 1, č. 3 [cit. 2015-04-26]. Dostupné z: http://www.ijrte.org/attachments/File/v1i3/C0272071312.pdf

[5] BISHOP, J. C#: návrhové vzory. Vyd. 1. Brno: Zoner Press, 2010, 323 s. ISBN 978-80-7413-076-2.

[6] TROELSEN, Andrew. ADO.NET Part III: The Entity Framework. Apress: Apress, 2012. ISBN 978-1-4302-4234-5. Dostupné z: http://link.springer.com/chapter/10.1007/978-1-4302-4234-5_23#

[7] What Is Windows Communication Foundation. Microsoft developer network. [online]. [cit. 2015-04-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx

[8] Twitter Bootstrap Part 1: What is Bootstrap Anyway?. Untame. [online]. 13.6.2012 [cit. 2015-04-23]. Dostupné z: http://untame.net/2012/07/twitter-bootstrap-part-1-what-is-bootstrap-anyway/

[9] ASP.NET MVC 4 Content Map. Microsoft developer network. [online]. [cit. 2015-04-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/gg416514%28v=vs.108%29.aspx

[10] CanvasXpress. CanvasXpress [online]. 2015 [cit. 2015-06-19]. Dostupné z: http://canvasxpress.org/

[11] Arbor.js. Arbor.js [online]. 2011 [cit. 2015-06-19]. Dostupné z: http://arborjs.org/

[12] The Open Graph Viz Platform. Gephi. [online]. [cit. 2015-04-23]. Dostupné z: http://gephi.github.io/

Seznam obrázků

Obrázek č. 1 Systém VisioProject ..... *- 2 -*
Obrázek č. 2 Systém WorkBook ..... *- 3 -*
Obrázek č. 3 Systém JIRA ..... *- 3 -*
Obrázek č. 4 Systém PayPanther ..... *- 3 -*
Obrázek č. 5 Fáze projektu ..... *- 5 -*
Obrázek č. 6 RUP artefakty ..... *- 6 -*
Obrázek č. 7 Zachmanův framework ..... *- 7 -*
Obrázek č. 8 Use-case diagram ..... *- 13 -*
Obrázek č. 9 Souhrn projektů a práce s nimi ..... *- 14 -*
Obrázek č. 10 Vytváření projektu ..... *- 15 -*
Obrázek č. 11 3-vrstvá architektura ..... *- 16 -*
Obrázek č. 12 Vzor Strategy ..... *- 18 -*
Obrázek č. 13 Vzor Repository ..... *- 18 -*
Obrázek č. 14 Třídní diagram ..... *- 19 -*
Obrázek č. 15 Entity framework ..... *- 21 -*
Obrázek č. 16 Obrázek systému Requirements ..... *- 23 -*
Obrázek č. 17 Obrázek systému Dashboard ..... *- 23 -*
Obrázek č. 18 Obrázek systému Projects ..... *- 24 -*
Obrázek č. 19 Responsivní prostředí ..... *- 25 -*
Obrázek č. 20 Mřížka bootsrapu ..... *- 25 -*
Obrázek č. 21 MVC ..... *- 26 -*
Obrázek č. 22 Orientovaný graf z CanvasXpress ..... *- 28 -*
Obrázek č. 23 Nerientovaný graf z CanvasXpress ..... *- 28 -*
Obrázek č. 24 Ukázka grafu generovaného pomocí Gephi ..... *- 29 -*

Seznam příloh

Součástí bakalářské práce je CD/DVD, obsahující program a dokumentaci.